UNIVERSIDAD CARLOS III
ESCUELA POLITÉCNICA SUPERIOR

GRADO EN INGENIERÍA INFORMÁTICA
TRABAJO DE FIN DE GRADO

DEFIDNET. Plataforma para la evaluación de redes de ciberdefensa.

AUTORA: Marta Canes López

TUTOR: Sergio Pastrana Portillo

Septiembre de 2014

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

En primer lugar, quiero agradecer a Sergio el magnífico tutor que es. Muchas gracias por haber confiado en mí para realizar este proyecto, por haberme ayudado en todos y cada uno de los problemas que he tenido, por haberme contagiado tu positivismo y por haber pasado alguna que otra tarde conmigo sin comer solucionando problemas sin solución. Estoy convencida de que no podría haber elegido un tutor mejor.

Quiero agradecer también, por supuesto, a mi familia el haberme aguantado no sólo durante este proyecto, sino durante toda mi vida; apoyándome y ayudándome a crecer como persona durante 23 años. A mis padres, que se han esforzado tanto para hacer que yo este aquí. A mis abuelos, las personas más adorables que jamás he conocido, ojalá algún día llegue a ser como ellos. Y a mi hermana Noelia, a la que, por muchas discusiones que tengamos, quiero con todo mi corazón.

A mis amigos, que pese a todo siempre siguen ahí, que aguantan mis largas ausencias y me esperan pacientemente. Como somos un grupo bastante disperso y probablemente me olvide de algunos nombres, simplemente mencionaré dos: Andrea y Erika. La primera por ser la única persona con la que no necesito hablar para sentir ese vínculo especial; y la segunda, por demostrar que a pesar de ser personas totalmente distintas ya no podría vivir sin ella.

A los chicos de la uni, compañeros y amigos durante tanto tiempo ya. Algunos que ya marcharon y otros que permanecen: Patricia, Geoconda, Natalia, Ainhoa, Miguel, Javi, Ana, Sergio, Alex, Epi, Dani, Luis, etc. Por todas las comidas, prácticas, quedadas y estrés en los exámenes juntos.

A mi jefe y compañeros de beca, que me han mimado durante lo que van a ser casi dos años y me han permitido tomarme todo el tiempo del mundo para realizar este trabajo, tiempo sin el cual seguramente todo hubiese ido mucho peor. A Harold por ser una fuente inagotable de conocimiento y permitirme más de lo que me merezco, a Ana por ser la alegría personificada y a Saúl por ser mi profesor particular de informática y de la vida en general.

Y por último, a Sebas. Amigo desde el inicio de esta aventura llamada universidad y pareja en este último tramo. Gracias por estar ahí siempre, desde el día que te conocí. Podría decir muchas cosas, pero creo que se me haría corto el papel. Así que solo diré que muchísimas gracias. Muchísimas gracias por ser tú.

Resumen

Hoy en día la red almacena cantidades ingentes de información. Tanto las grandes empresas y organizaciones como los propios gobiernos hacen uso de las extendidas facilidades ofrecidas por las modernas tecnologías, como son el almacenamiento en la nube o la computación ubicua, por ejemplo. Estos datos almacenados en el ciberespacio contienen información valiosa, y es por ello que constituyen un objetivo atractivo para potenciales atacantes. Es por lo tanto necesario dotar a las organizaciones de mecanismos y procedimientos que permitan garantizar la integridad, confidencialidad y disponibilidad de los datos.

La ciberseguridad es un conjunto de técnicas, herramientas y mecanismos que tienen como objetivo la protección de los sistemas, datos y usuarios que interaccionan en el ciberespacio. Hay numerosas herramientas de seguridad, como los conocidos cortafuegos, los registros de eventos, o los Sistemas de Detección de Intrusiones (*Intrusion Detection Systems*, IDSs). Éstos analizan el tráfico entrante y buscan cualquier signo de intento de comprometer la seguridad de los sistemas y redes. Si bien estos sistemas han sido eficientes en las últimas décadas, en la actualidad se requiere de medidas de mayor complejidad para hacer frente a las nuevas amenazas. En este sentido, las redes de detección de intrusiones (*Intrusion Detection Networks*, IDNs), combinan una serie de nodos distribuidos con distintas funcionalidades que se comunican entre sí intercambiando información.

Las IDNs tienen una capacidad de detección más amplia, gracias a su carácter colaborativo, por lo que permiten detectar ataques más sofisticados como son los ataques distribuidos, usualmente ocurriendo en zonas geográficas muy alejadas entre sí, y que en muchos casos no parecen tener relación entre ellos si no se analizan los datos de manera colaborativa y conjunta.

Las IDNs son una de las primeras barreras de seguridad que protegen los sistemas. Es por esta razón que los propios nodos que componen las IDNs son objetivo de los atacantes, quienes intentarán evadir la detección o disminuir la capacidad de respuesta. Se hace fundamental por lo tanto el diseño de IDNs robustas, que mantengan un mínimo nivel de protección incluso en caso de ser atacadas. En muchos casos, este diseño no es trivial y puede hacer perder mucho tiempo de gestión y mantenimiento a los administradores de seguridad.

El presente Trabajo de Fin de Grado expone el desarrollo de DEFIDNET (*DEFense of Intrusion Detection NETworks*), una aplicación para el diseño de IDNs que integra en un único proceso distintas herramientas externas mediante una interfaz gráfica sencilla e intuitiva. La aplicación toma como punto inicial un modelo diseñado previamente y un prototipo de la implementación. Mediante esta nueva interfaz se aúnan los distintos componentes y funcionalidades que permitía realizar el DEFIDNET original y se añaden algunas nuevas que no estaban previstas en el diseño inicial, incluyendo las que requieren el uso de herramientas o programas externos (como la exportación de imágenes de la IDN o la visualización de información en gráficos) en un proceso único y coherente que permite ahorrar tiempo en el diseño y gestión de las IDNs.

Abstract

Nowadays, information shared in the network has grown considerably. Both big corporations and companies, and even the governments, use the extended facilities offered by the new computing paradigms, such as the cloud computing or the ubiquitous computing. The information contained in the data is of great value, and therefore it constitutes an attractive objective for potential adversaries. It is thus required to provide the organizations with proper security and protection mechanisms in order to assure the confidentiality, integrity and availability of the data.

Cybersecurity is the ensemble of techniques, tools and methods whose aim is the protection of systems, data and users interacting in the cyberspace. There are many security tools, like the well-known firewalls, sensor logs or the Intrusion Detection Systems (IDSs). These systems analyze the input traffic and look for any sign of intrusion. While these systems have been widely effective and efficient in the last few decades, nowadays they are no longer appropiate for the detection of new threats. In this scenario, the Intrusion Detection Networks (IDNs) combine various nodes distributed among the protected network. These nodes share information, and may have different responsibilities and functions.

The IDNs are more sophisticated to detect attacks than traditional IDSs due to its cooperative nature. They are able to detect complex attacks, like distributed attacks happening at the same time in geographically separated areas, which seem

to be independent processes but have a common goal.

IDNs constitute one of the first security barriers protecting the systems and networks. This make themselves a target for potential attackers, who will try to evade the detection or bypass the security. It is thus basic to provide a robust design for IDNs in order to maintaint a certain level of protection even if some of the nodes are under attack or compromised. In many cases, this is a tedious task where security administrators must waste many time.

This project exposes the development of DEFIDNET (*DEFense of Intrusion Detection NETworks*), a graphical application that helps in the design of IDNs. The application integrates in a single graphical user interface many external tools. It takes as starting point an already developed model and an initial propotype. With this new interface, all the components and funcionalities from this initial model are integrated. It includes some new ones that were not included in the first design, including those that require the use of external tools (e.g. for visualizing images and graphics). The final product aims to facilitate the design of IDNs for security administrators, which turns into a save of resources and time.

Capítulo 1

Introducción y objetivos

1.1. Introducción

La ciberseguridad tiene como principal objetivo la protección de la información frente a errores, desastres naturales y ataques (tanto voluntarios como involuntarios), de tal manera que la probabilidad de que se produzca una de las circunstancias anteriores sea la menor posible y, en el caso de que se produzca, los daños se vean minimizados al máximo.

La seguridad es un tema sumamente importante en el ámbito de la informática, donde cantidades ingentes de información son enviadas, tratadas y utiiberlizadas. Internet es un gran almacén de información. Cada vez más, las grandes empresas e instituciones (así como el usuario medio) suben sus datos a la red como parte del proceso de almacenamiento en la nube[1], incluyendo información confidencial. Y de la misma manera que toda esa información es utilizada, también está expuesta a un gran número de amenazas, como describen las figuras 1.1 y 1.2 obtenidas del estudio realizado por Ponemon Institute [30] en 2013 y que muestran la distribución de ataques y la pérdida económica que estos suponen respectivamente.

[1] Almacenamiento en Internet.

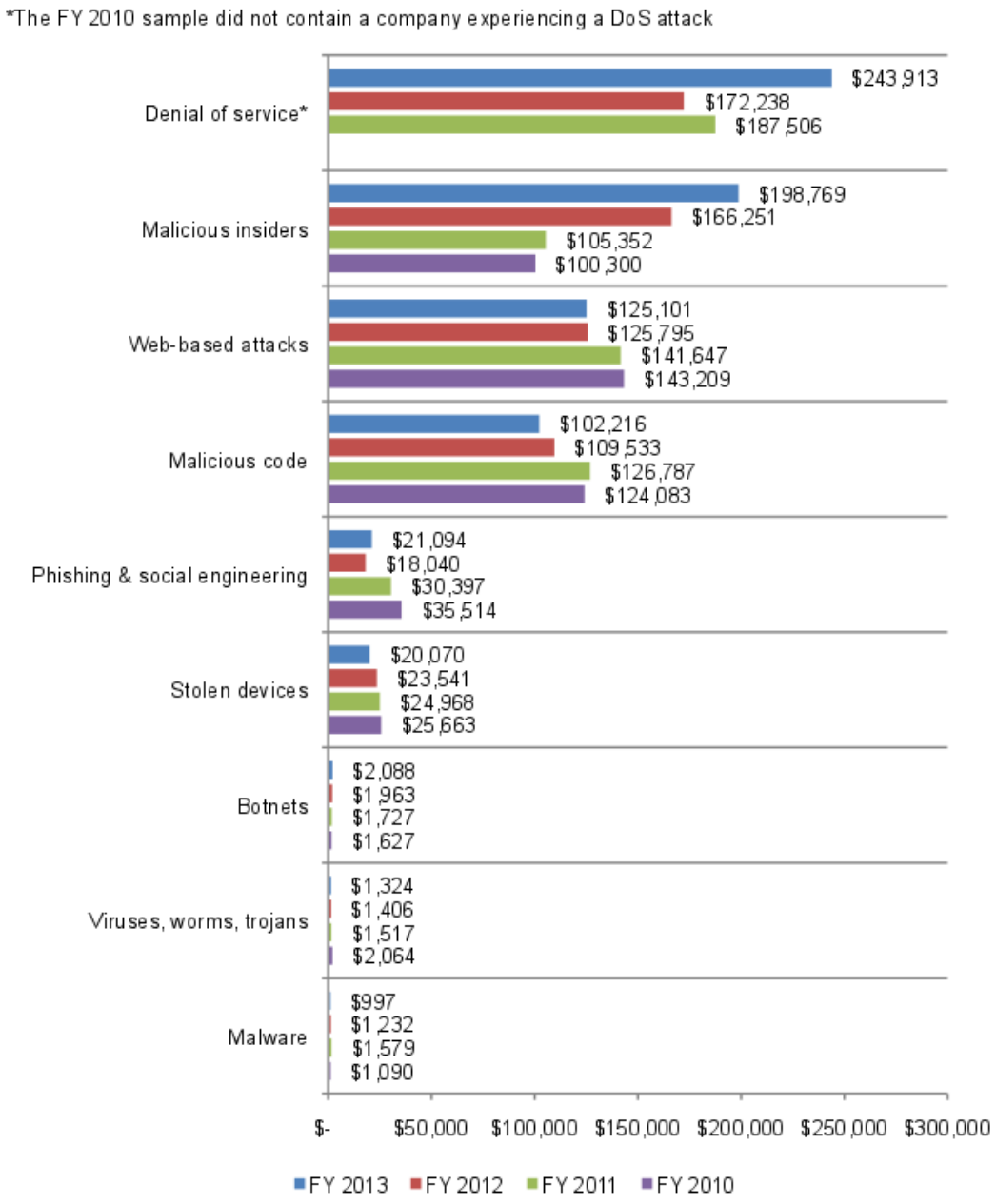

Figura 1.1: Estudio de costes para cada tipo de ataque. Ponemon Institute 2013.

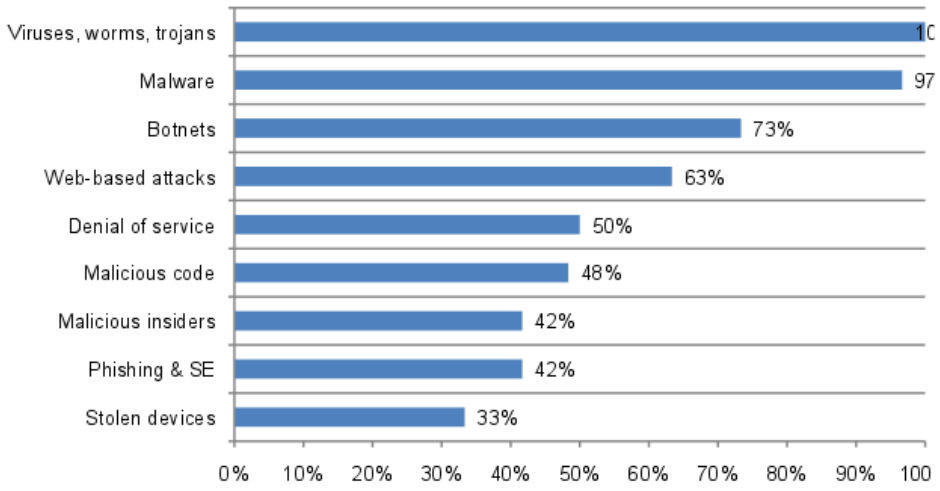

Figura 1.2: Estudio de tipos de ataque. Ponemon Institute 2013.

La información alojada en la red debe estar protegida, puesto que es vulnerable a ataques. Por ello, disponer de herramientas que permitan hacer más sencilla la tarea de proteger la información manejada en el ciberespacio es primordial.

Los Sistemas de Detección de Intrusiones, o IDSs por sus siglas en inglés (*Intrusion Detection Systems*), son programas utilizados comúnmente para analizar los accesos a un determinado *host*[2] y detectar posibles ataques a éstos, siendo una herramienta de seguridad ampliamente utilizada.

Los IDSs analizan el tráfico en busca de actividad intrusiva, bien paquetes de red (en cuyo caso se denominan *Network* IDS o NIDS) o bien registros locales de las máquinas protegidas, en cuyo caso se denominan *Host* IDS o HIDS. Típicamente el análisis se ha realizado de dos formas distintas. La detección basada en firmas busca patrones conocidos de ataques en la actividad monitorizada, mientras que la detección basada en anomalías primero modela la actividad que es considerada

[2] *Host*. Computador conectado a una red.

normal y luego considera como potencial intrusión cualquier cosa que se desvíe de este modelo. Si el paquete recibido responde a uno de esos patrones de ataque, el IDS notifica la intrusión y/o activa una serie de medidas.

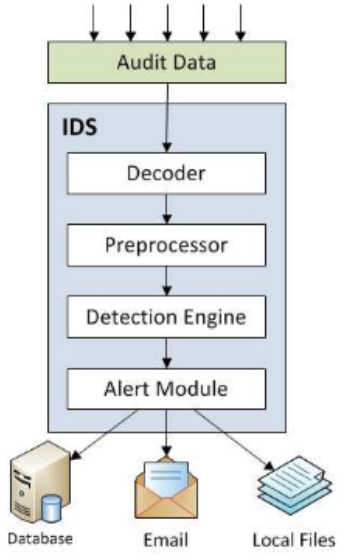

Figura 1.3: Arquitectura típica de un IDS.

Una Red de Detección de Intrusiones, o IDN por sus siglas en inglés (*Intrusion Detection Network*) está compuesta por un conjunto de IDSs distribuidos en la red protegida. Estas IDNs están pues compuestas de distintos nodos que actúan como IDSs individuales capaces de comunicarse con los demás elementos de la IDN, enviando y/o recibiendo información del resto. De esta manera, la IDN recibe información más detallada y a la vez global sobre los distintos accesos e intrusiones ocurridos, permitiendo la detección de ataques distribuidos.

Las IDNs son muy versátiles, puesto que permiten una gran cantidad de arquitecturas distintas. Las opciones de diseño son muy amplias, por lo que el usuario puede definir distintos tipos y analizar cual se ajusta más a los objetivos especificados.

DEFIDNET nace como una aplicación que permite diseñar y evaluar IDNs mediante una serie de herramientas para la definición de éstas y su posterior análisis. Básicamente, la aplicación permite definir distintas arquitecturas en la IDN (ver figura 1.4) y parametrizar los nodos participantes, por ejemplo para indicar su probabilidad de ser atacado o el coste de aplicar contramedidas en él. De esta manera, se pueden simular distintos escenarios que pueden darse en la vida real.

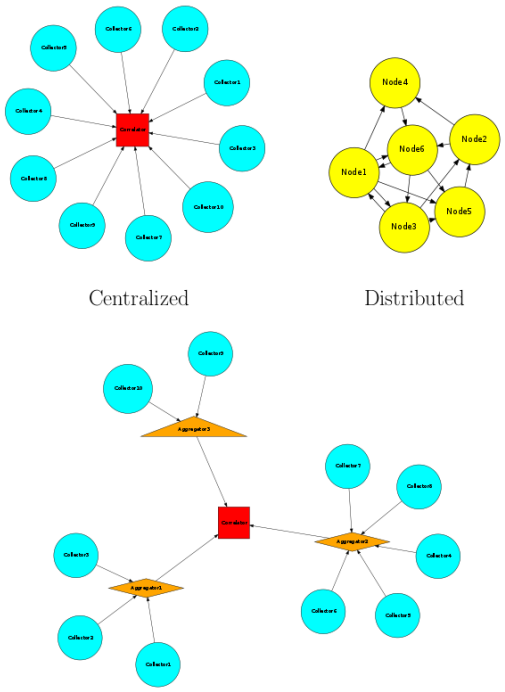

Figura 1.4: Distintos tipos de redes de detección de intrusiones.

1.2. Motivación

Analizando la situación actual en el entorno de la ciberseguridad, se observa que existe una gran cantidad de información acerca de los IDSs: sobre su definición, sus elementos, su funcionamiento, sus capacidades y demás características. Sin embargo, no se ha prestado tanta atención a la investigación y desarrollo de redes de detección de intrusiones capaces de proveer de seguridad en el altamente atacado ciberespacio

de hoy en día. Y las IDN son necesarias para proteger los sistemas, que cada vez están más distribuidos.

Las IDNs y los IDSs, por ser de las primeras barreras de seguridad con las que los atacantes se encuentran, son igualmente objeto de ataques, lo que supone un riesgo tanto para las propias ciberdefensas como para los sistemas y redes protegidos. Por tanto, el diseño de herramientas de ciberseguridad resistentes en presencia de ataques es fundamental.

No obstante, se pueden ver carencias en el mercado para cubrir las necesidades de diseño de IDNs y de búsqueda de la solución óptima a la hora de proteger los distintos sistemas frente a intrusiones y ataques no deseados. Actualmente, no existe en el mercado ninguna herramienta para facilitar el diseño y análisis de arquitecturas robustas de IDNs.

Dada su complejidad y tamaño, no está claro hasta qué punto es óptimo proteger cada nodo de la IDN, y por lo tanto se hace necesario contar con herramientas que faciliten el análisis de soluciones óptimas, que reduzcan el coste de su implementación y que mitiguen en mayor medida el riesgo de las IDNs. La motivación de la aplicación desarrollada en el presente proyecto proviene de la necesidad de contar con una herramienta útil y fiable que permita simplificar el proceso de diseño y análisis de IDNs, proporcionando información detallada sobre las mismas y obteniendo la solución óptima para su implementación en base al coste-beneficio, entendiendo beneficio como el riesgo mitigado.

1.3. Objetivos

El objetivo principal del presente Trabajo de Fin de Grado consiste en implementar una herramienta que sirva de ayuda para el análisis, diseño y gestión de grandes redes de detección de intrusiones, y que permita al usuario simplificar

la tarea de definición de IDNs efectivas, eficientes y robustas frente ataques. Para ello, se pretende complementar y ampliar el modelo inicialmente desarrollado por el tutor de este TFG como parte de su Tesis Doctoral [7], integrando funcionalidades ya existentes y añadiendo nuevas a través de una única interfaz gráfica que gestione la aplicación.

Así, los objetivos parciales definidos para el presente proyecto son:

- Estudiar la base teórica del problema y el estado del arte en el momento del desarrollo del proyecto para analizar las carencias existentes acerca del diseño de IDNs.
- Ofrecer una plataforma para la definición rápida y simplificada de una IDN, así como su posterior modificación y estudio para determinar la solución óptima coste-beneficio, integrando todas las funcionalidades necesarias en una única aplicación.
- Probar y evaluar el sistema resultante para verificar que su funcionalidad es la adecuada y que no existen problemas en su ejecución.
- Elaborar la documentación referente al proyecto, incluyendo el manual de usuario (*Anexo I*) y la presente memoria.

1.4. Alcance

Este proyecto parte de una base inicial que se puede consultar más detalladamente en la sección *2.4. Estudio del sistema inicial*.

Como se ha mencionado en el punto anterior, uno de los objetivos de este proyecto es completar la funcionalidad ya existente (adaptándola al entorno gráfico) y además proporcionar funcionalidad añadida en base a las características de la nueva interfaz gráfica, que permite una interacción usuario-aplicación mayor y más intuitiva.

La adaptación del código base requiere de numerosos cambios, en ocasiones complejos, para habilitar la interacción del usuario con la aplicación y la integración de los distintos componentes del modelo en un único módulo. En los siguientes apartados se expone una relación de las funcionalidades que se verán modificadas y adaptadas en la implementación del proyecto y también de las nuevas funcionalidades añadidas.

1.4.1. Funcionalidad adaptada

La funcionalidad ya existente en el código de partida de DEFIDNET y que será adaptada es:

- **Definición de una IDN**. Mientras que en el código de partida el usuario debe realizar las operaciones referentes a la definición de una IDN creando ficheros y modificando líneas manualmente, en esta nueva version de DEFIDNET se pretende que la aplicación guíe al usuario a través de formularios intuitivos y sencillos de entender.

- **Inserción, modificación y eliminación de nodos de la IDN**. Como en el punto anterior, estas acciones deben llevarse a cabo manualmente modificando ficheros, lo cual no es la manera óptima de proceder (es bastante fácil que el usuario cometa errores). La funcionalidad será adaptada para poder realizar dichas operaciones fácilmente mediante la principal nueva característica de la aplicación: el visualizador interactivo (o editor visual) de IDNs, que permitirá configurar una IDN dada de manera interactiva utilizando figuras.

- **Combinación de distintas IDNs en una IDN única**. También se aportará una interfaz gráfica para la selección de redes a combinar en una IDN de arquitectura *joined* y los nodos que actuarán como uniones entre ellas.

- **Generación de una solución y aplicación de contramedidas**. El proceso de generación de una solución se verá afectado para aprovechar las características del gráfico de resultados añadido a DEFIDNET y expuesto

en el siguiente punto. De la misma manera, la operación de aplicación de contramedidas utilizará dicho gráfico para que el usuario pueda interactuar seleccionando el punto deseado y asegurar la IDN en base a éste, reduciendo el riesgo con el menor coste asociado posible.

1.4.2. Funcionalidad añadida

Las funcionalidades añadidas como resultado de las posibilidades que ofrece la creación de una interfaz gráfica son:

- **Carga de una IDN**. Permite la carga en la aplicación DEFIDNET de una IDN ya existente, recuperando todas sus características. En el código inicial esta funcionalidad no tenía sentido, puesto que el usuario accedía a redes ya existentes a través de ficheros de sistema.

- **Exportación de una IDN**. En el código inicial no estaba integrada la exportación a imagen de una IDN (únicamente la generación de un fichero .dot, utilizado por la herramienta GraphViz para crear imágenes), sino que el usuario debía iniciar un nuevo proceso externo totalmente independiente del programa para poder generar la imagen. En la nueva DEFIDNET este proceso es interno de la aplicación.

- **Visualizador interactivo de IDN**. La principal nueva característica de DEFIDNET. Es un editor visual que permite modificar los atributos de la IDN mediante el uso de figuras que representan los nodos y conexiones. Es completamente interactivo.

- **Gráfico de los resultados**. Gráfico obtenido de la generación de una solución que muestra el Frente de Pareto[3] resultante y mediante el cual el usuario puede

[3]Conjunto de valores óptimos de una función objetivo de tal manera que representan el punto de equilibro en el que mningún elemento de la función puede mejorar su situación actual sin perjudicar al otro.

seleccionar un punto para aplicar las contramedidas necesarias para asegurar una IDN.

1.5. Estructura de la memoria

Este documento se divide en ocho bloques de información, los cuales se dividen a su vez en secciones más específicas sobre el tema a tratar en cada bloque individual, de forma que la información es fácilmente accesible y se encuentra organizada de manera coherente y efectiva.

Para facilitar la lectura y el uso de la presente memoria, a continuación se especifican las distintas partes que la componen.

- **Capítulo 1. Introducción y objetivos.** En este capítulo se realiza una breve descripción del problema, así como la motivación para llevar a cabo el proyecto. Además, se especifican los objetivos que se pretenden conseguir con la aplicación DEFIDNET y el alcance de ésta, analizando la funcionalidad replicada y la funcionalidad añadida.
- **Capítulo 2. Estado del arte.** En esta segunda parte se realiza una breve aproximación a los conceptos teóricos necesarios para la comprensión del problema, puesto que forman la base teórica sobre la que se sustenta la aplicación DEFIDNET. Se analiza el estado actual respecto al problema, indagando en las herramientas alternativas ya existentes en el mercado y los posibles destinatarios, potenciales usuarios de la aplicación. Se hablará sobre ciberseguridad, IDSs, IDNs y sus características. Se pretende un acercamiento inicial y restringido a la cantidad de información existente, pero que cubra los conceptos básicos.
- **Capítulo 3. Gestión del proyecto.** En este capítulo se abordan los temas referentes a la planificación del proyecto, estimando la duración del mismo, así como su coste económico. Además, se indica la metodología de desarrollo,

explicando el por qué de su elección, aportando ventajas y desventajas frente a otras opciones posibles. Además, se habla brevemente del control de versiones llevado a cabo.

- **Capítulo 4. Análisis del problema.** Apartado en el que se recogen los requisitos obtenidos, los cuales deben cumplirse para que la aplicación implemente las funcionalidades, restricciones y demás cuestiones especificadas por el cliente y derivadas. Así mismo, se recogen los casos de uso que permiten describir las secuencias de operaciones que se llevarán a cabo en la aplicación una vez esté operativa y que sirven de base para la obtención de requisitos.

- **Capítulo 5. Diseño.** En este capítulo se explican las decisiones tomadas en cuanto a diseño, ofreciendo mediante diagramas y texto una visión clara de la arquitectura elegida y de como está estructurada la aplicación.

- **Capítulo 6. Implementación.** La sección de implementación describe el proceso llevado a cabo para la codificación de la aplicación DEFIDNET, siendo el apartado con más aspectos técnicos de la presente memoria y que ofrece más información sobre el trabajo llevado a cabo en este Trabajo de Fin de Grado en el día a día. De la misma manera, se explican brevemente las características de las distintas herramientas externas utilizadas.

- **Capítulo 7. Pruebas y evaluación del sistema.** Capítulo dedicado a la recolección de las pruebas realizadas sobre la aplicación, mostrando los resultados obtenidos y la evaluación final del sistema una vez éste se encuentra en estado finalizado.

- **Capítulo 8. Conclusiones y líneas futuras.** Se exponen las opiniones acerca del proyecto, así como las ideas obtenidas y estado final de la aplicación. Así mismo, se consideran posibles formas de continuación del proyecto para añadir funcionalidad y/o mejorar la ya existente.

- **Glosario.** Tras el cuerpo del documento, se presenta un glosario de siglas y

términos que permite comprobar rápidamente el significado de las distintas palabras técnicas y acrónimos utilizados.

- **Bibliografía.** Se recoge la bibliografía consultada durante el desarrollo del proyecto y la redacción de esta memoria.
- **Anexo I. Manual de usuario.** Al final de este documento se puede encontrar el manual de usuario, con información sobre el uso de DEFIDNET y cuestiones técnicas acerca de la aplicación.

Capítulo 2

Estado del arte

2.1. ¿Qué es la ciberseguridad?

La ciberseguridad se define como los procedimientos enfocados a la protección de la información almacenada en el ciberespacio [1]. Su principal objetivo es mantener segura esa información, protegiéndola de errores, fallos humanos, desastres naturales, ataques intencionados y demás eventos que puedan atentar contra la integridad o confidencialidad de los datos.

La ciberseguridad es una necesidad en la actualidad, puesto que en Internet se almacena información confidencial y en ocasiones sumamente relevante. Por ello, es de vital importancia contar con sistemas de protección que reduzcan las posibilidades de sufrir una situación de peligro y que minimicen los impactos producidos sobre el sistema en el caso de que no haya sido posible evitarla.

Los ataques por parte de personas malintencionadas son cada vez más modernos y sofisticados. Esto hace que sea difícil prevenirlos, detectarlos y minimizar los daños que causan. La ciberseguridad debe avanzar de manera paralela a estos nuevos ataques para realizar su función correctamente. Por esta razón, la ciberseguridad

[1] Ámbito de comunicaciones constituido por una red informática.

cada día cuenta con más sistemas y herramientas que permiten la protección de la información.

2.2. Sistemas de Detección de Intrusiones

Los Sistemas de Detección de Intrusiones, o IDS por sus siglas en inglés (*Intrusion Detection Systems*) son programas cuya función principal es detectar posibles ataques a un *host* mediante accesos no autorizados a éste. Para ello, los paquetes recibidos son filtrados por el IDS, que los analiza buscando posibles ataques. El objetivo final de cualquier IDS es detectar intentos de comprometer la seguridad de los sistemas, con el fin de tomar las medidas de seguridad correspondientes o modificar las existentes.

A continuación se pasan a definir los principales tipos de IDS según varias categorías.

Existen distintos tipos de IDS según el ámbito del tráfico que analizan:

- **NIDS (*Network IDS*)**. El IDS escanea los paquetes de red a nivel de enrutador o *host*. Este tipo de IDS se ha vuelto muy popular en los últimos años debido al crecimiento desmesurado de internet y a su rapidez a la hora de detectar intrusiones [26]. Normalmente, utiliza un método de detección basado en firmas, explicado en la siguiente clasificación.
- **HIDS (*Host IDS*)**. El IDS analiza los registros locales del host. Éstos, consultan diferentes tipos de registros de archivos (*kernel*, sistema, servidores, red, cortafuegos, y más) y los comparan con una relación de patrones de ataque conocidos almacenados en una base de datos interna [26].

Los IDS también pueden clasificarse según el método de detección de intrusiones empleado, siendo las formas expuestas a continuación las más comunes y empleadas:

- **Detección basada en firmas o patrones de ataque**. Mediante este método el IDS detecta posibles ataques mediante firmas; es decir, patrones de ataque. El IDS dispone de una base de datos que contiene el *modus operandi* de una serie de ataques conocidos. Éste, analiza los paquetes recibidos buscando comportamientos que coincidan con alguno de los registrados como ataques. El éxito de la detección depende del número de registros en la base de datos y de lo actualizados que estén. El problema de este método es que no puede detectar ataques nuevos o lo suficientemente recientes que aún no están contemplados en la base de datos [7].

- **Detección basada en anomalías**. El IDS basado en la detección de anomalías trabaja de una manera muy distinta al basado en detección de firmas. En este caso, el IDS busca desviaciones en los patrones considerados normales. Para ello, primero modela la actividad considerada como normal y después considera las desviaciones del modelo creado como ataques o accesos no autorizados. De este modo, es capaz de capturar ataques nuevos o de reciente creación que el método anterior era incapaz de detectar. El problema al que se enfrenta este método son los *falsos positivos*, puesto que es muy fácil registrar comportamientos inusuales que realmente no son ataques [7].

2.3. Redes de detección de intrusiones

2.3.1. ¿Por qué el uso de IDNs?

Los IDSs cumplen eficientemente sus funciones hasta cierto punto. Sin embargo, el trabajar de manera aislada deriva en una dificultad cada vez mayor para detectar nuevos tipos de intrusiones. Por ejemplo, los ataques distribuidos en distintos puntos de una red son difíciles de detectar puesto que los IDSs que se encargan de protegerla no se comunican entre sí y no utilizan información común.

Una Red de Detección de Intrusiones (IDN) es una red colaborativa de nodos, cada uno implementando una determinada función, que colaboran para trabajar juntos activamente. Su función es suplir las carencias de los IDSs aislados compartiendo información entre los distintos nodos, de tal manera que ganen experiencia y conocimiento de los ataques producidos a cualquier punto de la red. De esta manera, cada nodo recolecta información del resto y envía los datos recolectados por sí mismo a los demás.

Un *falso positivo* (alarma generada para un evento de no intrusión) produce costes de los recursos humanos utilizados para investigar el caso, mientras que un *falso negativo* (intrusión no detectada) puede poner en peligro el sistema. Si el sistema de decisión es demasiado sensible puede traer consigo numerosos *falsos positivos* y si el sistema es conservador puede tener un ratio de *falsos negativos* muy alto [5]. Por tanto, es sumamente importante apurar la sensibilidad mediante diseños efectivos que balanceen al máximo el sistema de decisión de la IDN.

2.3.2. Nodos

Los nodos son los distintos sistemas individuales que forman una IDN. Cada nodo o grupo de nodos pueden desempeñar roles distintos que se complementan para una mayor protección de la red frente a ataques.

Siguiendo el modelo conceptual aportado por Pastrana [7] un nodo se compone de distintos canales. Estos canales representan la entrada y salida de información. Así mismo, cada tipo de nodo puede ejecutar unas determinadas funciones.

Para explicar correctamente los distintos canales de un nodo es necesario definir primero los tres tipos de mensajes que éste gestiona:

- **Mensajes de detección de intrusiones.** Los mensajes de detección de intrusiones, o IDMsg por sus siglas en inglés (*Intrusion Detection Messages*),

representan cualquier mensaje que contenga información sobre la detección de ataques.

- **Eventos locales.** Los mensajes del tipo LE (*Local Events*) son los mensajes relacionados con la monitorización de información local del *host*. También, incluyen la información del tráfico de red detectada por los sensores de manera local.

- **Acciones responsivas.** Las acciones responsivas, o RA (*Response Actions*), se producen cuando se detecta una intrusión.

Una vez definidos los tipos de mensajes que los distintos nodos de una IDN envían o reciben, se definen los canales de cada nodo individual divididos en las categorías de entrada y de salida:

- **Canales de entrada.**

  - **Canal de entrada de mensajes de detección de intrusiones.** El canal IIDM (*Input Intrusion Detection Messages*) recibe los mensajes del tipo IDMsg de otros nodos de la IDN.
  - **Canal de eventos locales**. El canal LE (del inglés *Local Events*) recibe la información recolectada localmente.

- **Canales de salida.**

  - **Canal de salida de mensajes de detección de intrusiones.** El canal de salida de mensajes de detección de intrusiones, o OIDM (*Output Intrusion Detection Messages*), es utilizado para enviar mensajes IDMsg a otros nodos.
  - **Canal de acciones responsivas**. El canal RA (*Response Actions*) activa las medidas y acciones en respuesta a las intrusiones detectadas.

Así mismo, los nodos pueden ejecutar cuatro funciones para comunicarse con el resto, listadas a continuación:

- **Función de compartición de eventos.** La función ESF (*Event Sharing Function*) se encarga de la comunicación entre distintos nodos. En primer lugar, procesa los mensajes IDMsg entrantes a través del canal IIDM y envía esa información a la función de detección distribuida, o DDF. En segundo lugar, procesa y formatea la salida de las funciones DDF y LDF para enviar mensajes a través del canal OIDM.

- **Función de detección local.** La función de detección local o LDF (*Local Detection Function*) utiliza la información de los eventos locales y del tráfico de red para realizar la detección local. La información recolectada por esta función es utilizada por las tres funciones restantes: por la función de compartición de eventos (ESF) para compartir la información con otros nodos, por la función de detección distribuida (DDF) para correlar los datos locales con otros IDMsg y por la función de respuesta (RA) si es necesario aplicar acciones.

- **Función de detección distribuida.** La función de detección distribuida, o DDF por sus siglas en inglés (*Distributed Detection Function*), recibe información de las funciones LDF y ESF y después la correla para enviar los datos a la función de respuesta: RF en caso de que se necesite aplicar acciones o a la función ESF si requiere de comunicación con otros nodos.

- **Función de respuesta.** La funcion RF (*Response Function*) es activada cuando una acción de respuesta es requerida.

En la figura 2.1 se puede ver un esquema de los canales y funciones de un nodo.

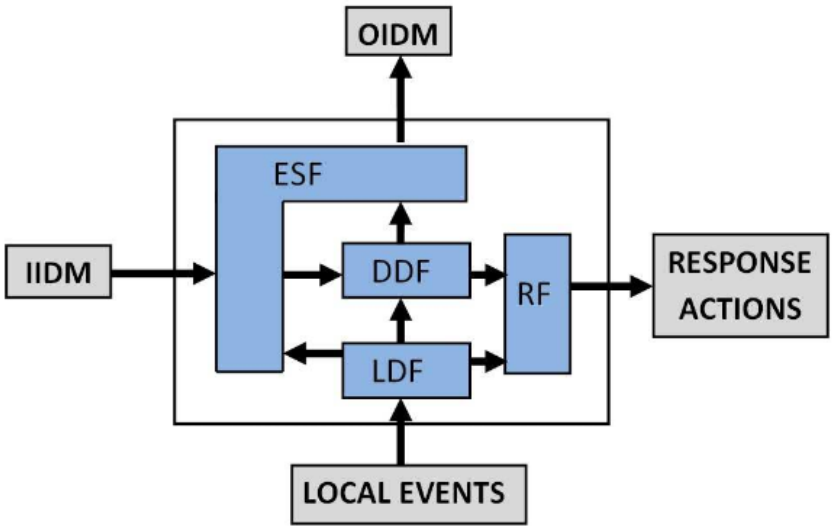

Figura 2.1: Canales y funciones de un nodo.

Los nodos pueden tener distintos roles en una IDN dependiendo de los canales que tengan activos y de las funciones que realicen dentro de ésta. En los siguientes puntos se habla de dichos roles y sus principales características.

- **De detección local**. Los LD (*Local Detection*) son, como su propio nombre indica, nodos que detectan intrusiones basándose en el análisis de eventos locales únicamente. Si bien este tipo de nodo es explicado para completar la información aportada, no es utilizado en las IDNs, puesto que no puede enviar o recibir información con otros nodos, y por lo tanto no puede pertenecer a la red de detección.

- **De detección local y compartición de alertas**. Los nodos del tipo LDA (*Local Detection and Alert Sharing*) son los más complejos, puesto que usan todos sus canales. Estos nodos utilizan los eventos locales y la información recibida de otros nodos para detectar posibles ataques. Además, los LDA comparten al mismo tiempo información con los otros nodos de la IDN.

- **De correlación pura**. Este tipo de nodos, llamados PC (*Pure Correlation*)

reciben información de otros nodos y la correlan, decidiendo qué deben hacer en base a los datos recibidos (pudiendo generar una respuesta). Estos nodos no utilizan los eventos locales para detectar intrusiones.

- **De correlación remota**. Los nodos de correlación remota, RC por sus siglas en inglés (*Remote Correlation*), reciben información de otros nodos, la correlan y después envían los resultados obtenidos a otros nodos. Los RC no generan ninguna respuesta.

- **De correlación y detección remotas**. Los nodos RCD (*Remote Correlation and Detection*) utilizan los eventos locales y la información recibida de otros nodos para detectar intrusiones. Seguidamente, envían la información correlada a otros nodos, sin generar respuestas.

- **Recolector de datos**. Los DC (*Data Collector*) detectan accesos no autorizados a través de eventos locales y tráfico monitorizado en su entorno, y envían la información recolectada a otros nodos.

2.3.3. Arquitecturas de IDNs

Las IDNs, como se ha comentado con anterioridad, conectan distintos nodos que comparten información. Dependiendo de las conexiones y funcionalidades de estos nodos pueden definirse distintas arquitecturas. A continuación se describen las más comunes, ampliamente aceptadas por la comunidad, que son las que luego se han considerado en el diseño de DEFIDNET.

2.3.3.1. Arquitectura centralizada

Las IDNs de arquitectura centralizada constan de un nodo central encargado de recolectar y analizar la información recibida de los nodos circundantes, que están conectados a él. Esos nodos circundantes actúan como sensores de información que envían datos al nodo central. Éste agrega, filtra y normaliza la información recibida, y pone en marcha las medidas necesarias en caso de detectar una intrusión.

El problema de este tipo de IDNs es que el nodo central es un punto crítico, puesto que si es atacado toda la IDN se ve en peligro. Otro problema es que dicho nodo requiere de unas capacidades de procesamiento y comunicación mucho mayores a las del resto [7].

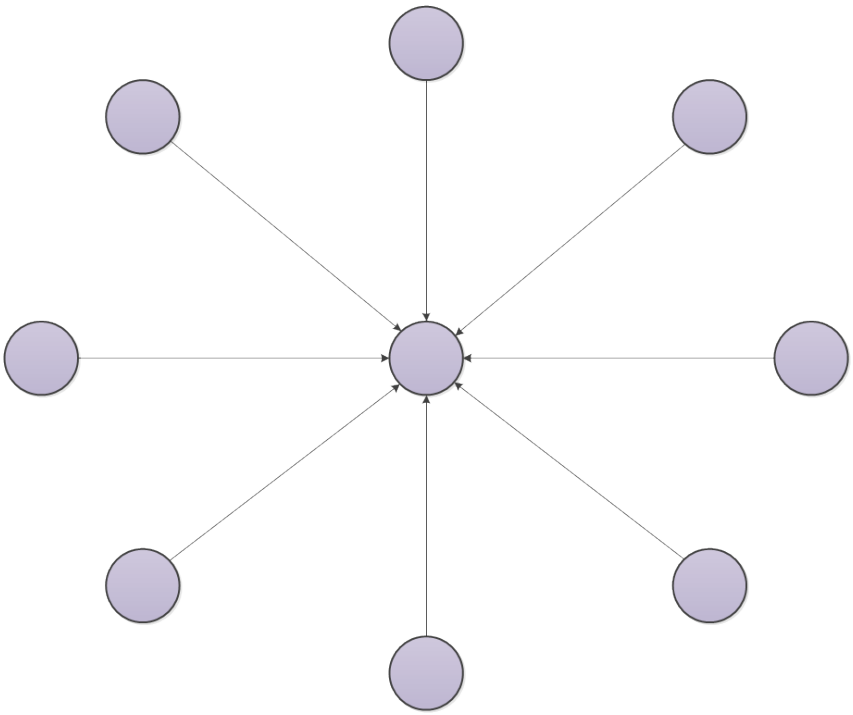

Figura 2.2: Arquitectura IDN centralizada.

2.3.3.2. Arquitectura distribuida

En este tipo de arquitectura los nodos tienen dos funciones. En primer lugar, realizar detección local basada en la monitorización del tráfico y de los eventos que se producen localmente. Y en segundo lugar, compartir información con el resto para correlarla y realizar una detección distribuida de ataques que no pueden ser detectados basándose únicamente en la información local.

Las IDN distribuidas pueden ser:

- **Totalmente distribuidas.** Los nodos que las conforman están conectados

todos con todos.

- **Parcialmente distribuidas.** Los nodos se conectan a un subconjunto del resto, determinando el tamaño de dicho subconjunto el porcentaje de conexión definido para la IDN.

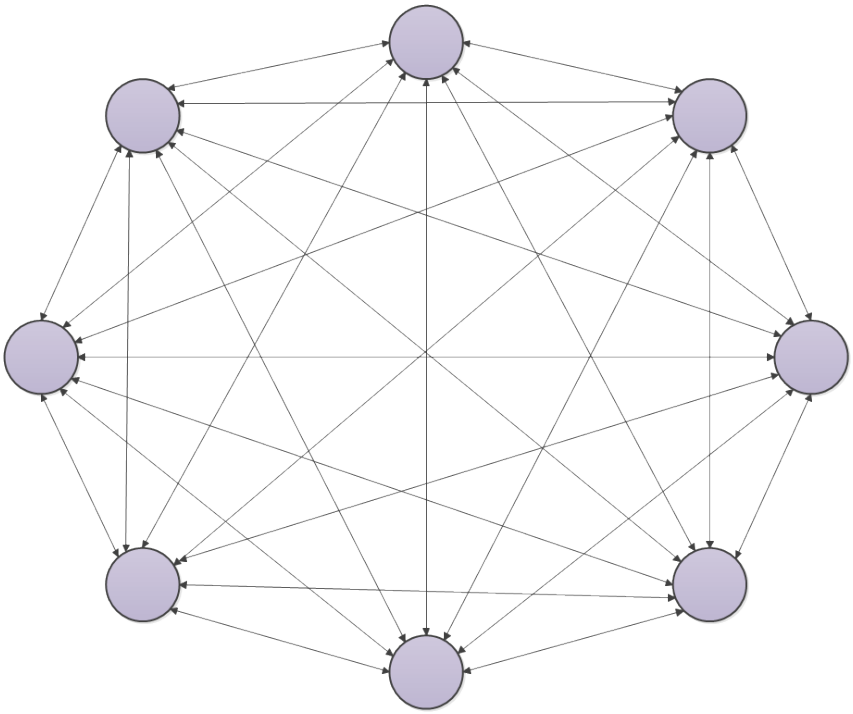

Figura 2.3: Arquitectura IDN totalmente distribuida.

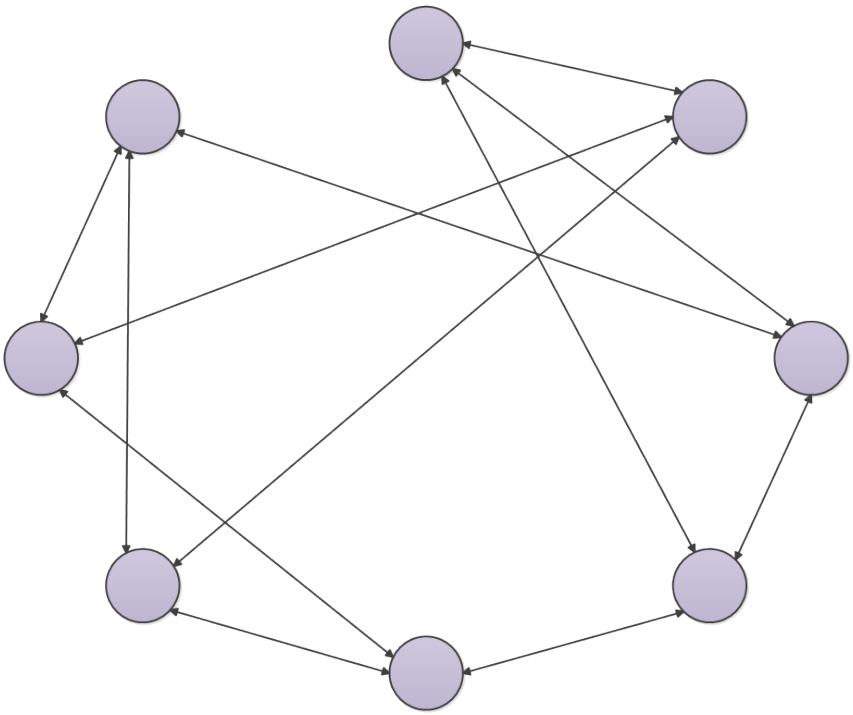

Figura 2.4: Arquitectura IDN parcialmente distribuida.

2.3.3.3. Arquitectura jerárquica

Las IDNs jerárquicas responden a la estructura de árboles. La IDN está organizada en distintos niveles que reciben información de los niveles inferiores.

Cada nivel tiene nodos con diferentes roles y responsabilidades sobre la detección de intrusiones de tal forma que se cubre toda la IDN. Cada nivel está dividido en zonas o *clusters*[2].

En cada *cluster*, sus miembros recogen información local y la traspasan al encargado del *cluster*, el cual filtra, agrega y normaliza la información recibida, y la envía a su vez al nodo padre (que se encuentra en el nivel inmediatamente superior

[2] *Cluster*. Conjunto de computadores.

por encima de ellos). Éste, analiza la información y realiza su función dependiendo de sus responsabilidades en el nivel [7].

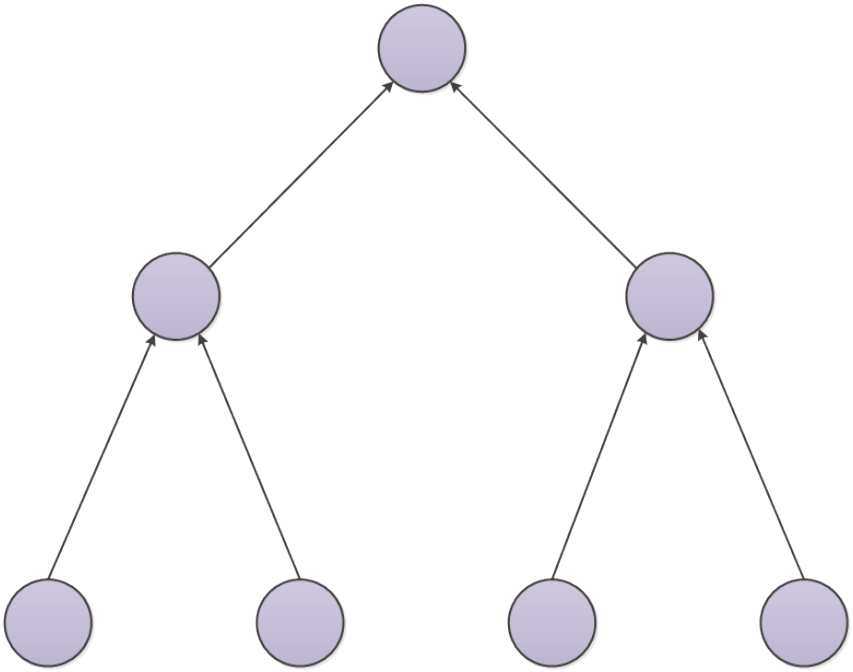

Figura 2.5: Arquitectura IDN jerárquica.

2.3.3.4. Arquitectura compuesta

Las IDN compuestas resultan de la combinación de varias arquitecturas distintas que no responden a un patrón concreto.

Un ejemplo de arquitectura compuesta o combinada puede ser la arquitectura *mixed-hc*, que une una IDN jerárquica con una centralizada, como muestra la figura 2.6. Otro ejemplo es la conjunción de una jerárquica con una en forma de anillo (*mixed-hr*).

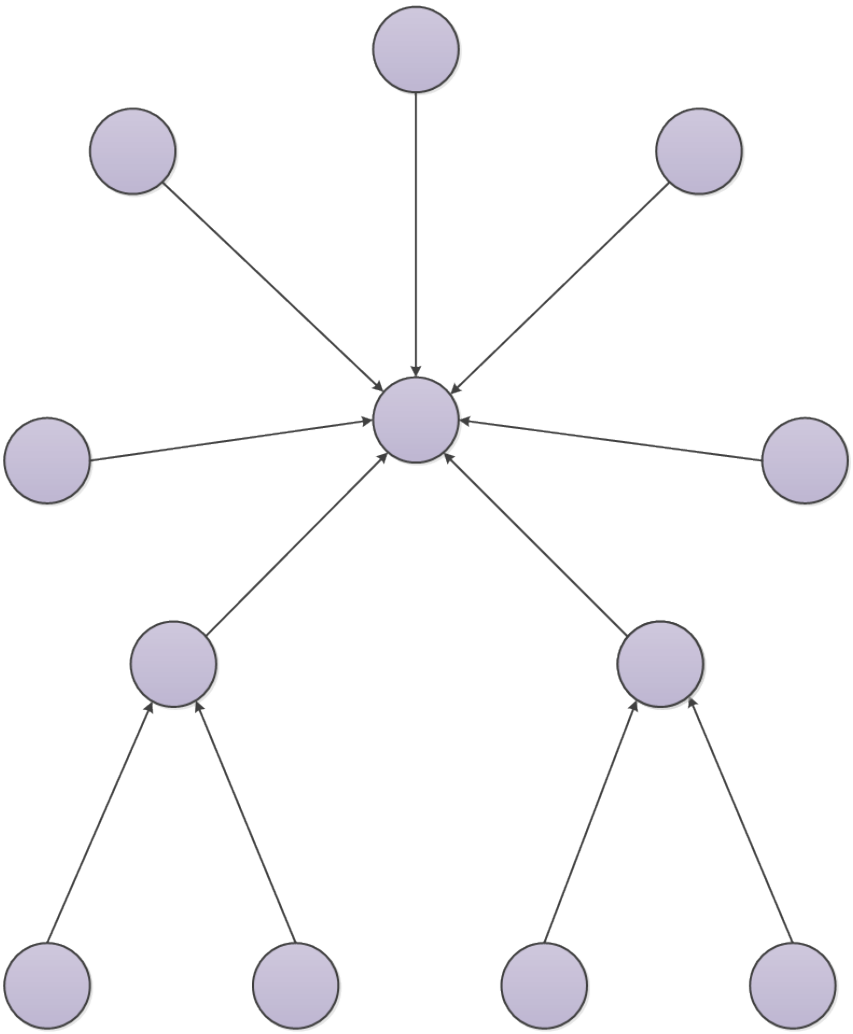

Figura 2.6: Arquitectura IDN compuesta.

2.3.4. Tipos de ataque

En este apartado se describen los tipos de ataques existentes. En primer lugar, se definen los ataques típicos a los canales de comunicación. Seguidamente, se realiza un estudio más detallado de los existentes contra IDSs.

La primera clasificación más general y conocida de ataques a canales de comunicación es la siguiente:

- **Intercepción**. Un tercero no autorizado es capaz de captar información enviada y recibida a través del canal de comunicación.
- **Modificación**. Situación similar a la anterior. En este caso, un tercero no solo accede a la información, sino que también la modifica y falsifica.
- **Interrupción**. Se corta el flujo de información entre el emisor y el receptor de un canal de comunicación.
- **Generación**. Un tercero genera información falsa y la introduce en el canal de comunicación.

Como se ha comentado previamente, recientemente la comunidad investigadora en el ámbito de la seguridad ha comenzado a valorar los posibles ataques que pueden producirse contra los IDSs. En un reciente trabajo [2], Corona y otros han establecido una taxonomía de ataques que se divide en 6 categorías dependiendo de su forma de actuación:

- **Ataque por evasión**. El ataque por evasión se basa en evitar que el IDS sea capaz de detectar el ataque que se lleva a cabo.
- **Ataque por sobreestimulación**. Mediante estos ataques el IDS recibe un gran número de ataques que responden a patrones de ataque. De esta forma, el IDS se ve desbordado por la cantidad de información recibida y la cantidad de alertas que debe generar y se colapsa.
- **Ataque por envenenamiento**. El envenenamiento consiste en el envío constante de patrones probablemente erróneos al IDS, de tal manera que la función de detección se va entrenando con esos patrones, lo que hace que el IDS genere firmas que alteran peticiones normales.

- **Ataque mediante denegación de servicio**. Los ataques de denegación de servicio tienen como objetivo desactivar o dañar la función de detección del IDS. Para ello, se intenta colapsar el servicio enviando un número muy elevado de paquetes que casan con las firmas especificadas en la base de datos del IDS.
- **Ataque mediante secuestro de respuesta**. El atacante genera alertas incorrectas que hacen que el IDS se confunda y bloquee conexiones válidas.
- **Ataque mediante ingeniería inversa**. El uso de la ingeniería inversa para atacar una IDN consiste en enviar determinados patrones estudiados a la IDN y ver como responde en cada caso. De esta manera, el atacante recolecta información sobre la IDN y puede llegar a descubrir las firmas que maneja en su base de datos.

Dependiendo del objetivo del atacante, los IDS pueden sufrir uno o varios de estos ataques en un momento determinado. Éstos causarán un impacto sobre los sistemas, el cuál dependerá tanto de la severidad del ataque como de la responsabilidad o rol del nodo atacado. Como se verá más adelante, en DEFIDNET estos impactos se podrán definir manualmente y permitirán al usuario tener conciencia del riesgo al que está expuesta la IDN manejada.

2.4. Estudio del sistema inicial

Este Trabajo de Fin de Grado desarrolla un nuevo proyecto completo utilizando como base un prototipo desarrollado por el Dr. Sergio Pastrana [7].

En un proyecto de estas características, en el cual se parte de un código inicial desarrollado con anterioridad, es importante realizar un estudio previo para permitir una integración rápida de los nuevos elementos y un desarrollo eficiente, evitando conflictos e incongruencias que puedan generar errores y fallos.

Mediante el estudio del sistema inicial se conocen y comprueban:

- **La funcionalidad existente**. Ésto permite especificar qué funcionalidad deberá ser añadida al código base y cuál deberá ser modificada para adaptarse a los objetivos del nuevo proyecto.
- **Los posibles errores**. Pueden existir errores y fallos en el programa base, por lo que el inicio del nuevo proyecto es el momento idóneo para solucionar esta clase de conflictos que a la larga pueden perjudicar enormemente el desarrollo de éste.
- **Las normas de codificación**. Se deben conocer las reglas de codificación utilizadas para crear código coherente y perfectamente integrado en estilo con el anterior.

En primer lugar, se realizó un examen superficial que permitió conocer la funcionalidad general de la aplicación. De este primer estudio se pudo obtener la relación de funcionalidades que deberían ser añadidas para alcanzar los objetivos fijados en el proyecto.

Después, se realizó un estudio más detallado y en profundidad para verificar el estado de la funcionalidad existente e identificar las modificaciones necesarias para adecuarla a la nueva interfaz gráfica y potenciar sus características aprovechando las ventajas del uso de una capa de presentación interactiva.

Se pueden consultar las funcionalidades modificadas y añadidas en los puntos *1.4.1. Funcionalidad adaptada* y *1.4.2. Funcionalidad añadida.*

2.5. Destinatarios

DEFIDNET ha sido concebida para ser utilizada por usuarios que por motivos profesionales o académicos requieran poder diseñar o emular IDNs reales y analizar

el coste-beneficio que cada tipo de configuración podría proporcionar a la IDN en cuestión.

Por tanto, esta herramienta está principalmente destinada a un usuario con conocimientos en el área de la ciberseguridad y más concretamente en IDSs e IDNs, puesto que estará familiarizado con los términos y expresiones utilizados y con los procedimientos a seguir. Así mismo, será capaz de introducir valores realistas y coherentes dado su conocimiento en el ámbito y los límites y restricciones de éste.

Sin embargo, también se considera una herramienta apta para un usuario con conocimientos básicos en la teoría tratada, si bien será necesaria una lectura completa y exhaustiva del manual de usuario y documentación proporcionada para poder utilizar de manera correcta la aplicación.

No se considera una herramienta provechosa para el usuario no iniciado en el diseño y análisis de IDNs (sin ningún tipo de conocimiento en la materia), puesto que requiere un esfuerzo de aprendizaje y de comprensión que no merecen la pena frente a los beneficios que puede ofrecer a este colectivo.

2.6. Situación actual

DEFIDNET es una aplicación de investigación que facilita el diseño de redes de ciberdefensa en auge como son las IDNs, pero sobre las cuales existen aún pocos estudios realizados. Una IDN se encarga de proteger una red de información, y por tanto, muchas veces se convierte en el blanco de los ataques para ser anulada y poder acceder a la red que protege.

Analizando el estado del arte y las distintas herramientas existentes se puede concluir que no existe ninguna destinada al diseño y análisis de IDNs resistentes a ataques comparable con DEFIDNET. Por tanto, ésta supone un paso hacia delante

en la implementación de IDNs seguras, siendo la única aplicación de la que se tiene constancia que permite analizar y generar soluciones óptimas para una IDN dada.

En esta sección se exponen las distintas alternativas tenidas en cuenta para el desarrollo de DEFIDNET en el presente proyecto.

El prototipo de DEFIDNET requería de una interfaz gráfica para aunar en un único proceso todas las opciones que proporciona, sin necesidad de utilizar programas externos que rompan el flujo natural de actividad. Y dentro de esa capa de presentación se requería como principal objetivo del proyecto un visualizador interactivo con el cual el usuario pudiera modificar una IDN dada y operar con ella mediante esquemas de figuras.

Se barajaron algunos simuladores de redes que podrían proporcionar el visualizador buscado. Algunos de esos simuladores de redes eran:

- **OMNeT++**. Es un framework para la simulación de redes basado en C++. Se utiliza para modelar sistemas con eventos discretos[3] [28]. Se puede consultar más información en la página web del programa: http://www.omnetpp.org/.
- **NS-2**. NS-2 es un simulador de redes basado en eventos discretos. Proporciona soporte para la simulación de numerosos protocolos y está escrito en C++ [19]. Página web del proyecto: http://www.isi.edu/nsnam/ns/.
- **GloMoSim**. Al igual que los dos anteriores, GloMoSim (*Global Mobile Information System Simulator*) es un software de simulación de protocolos de red que permite modelar sistemas mediante eventos discretos. Utiliza PARSEC, un lenguaje de simulación desarrollado por la universidad de California y basado en C [31].Para saber más sobre GloMoSim, visite: http://goo.gl/rPw9Cs.

[3]Evento cuyo estado cambia en instantes espaciados en el tiempo.

Tras el estudio de los simuladores citados, finalmente se optó por el desarrollo de una interfaz gráfica nueva utilizando diversas herramientas (véase la sección *2.7. Herramientas utilizadas*) con el fin de conseguir un grado de personalización muy superior al ofertado por las opciones arriba expuestas.

Esta decisión fue influenciada por la existencia de GraphPanel, una herramienta sencilla que se presentó como la base perfecta para el desarrollo del visualizador interactivo de DEFIDNET por las siguientes razones:

- Se requería un nivel de personalización muy alto del visualizador de IDNs para proporcionar todas las funcionalidades deseadas. Aunque las opciones barajadas permitían cierto grado de personalización, no podían competir contra GraphPanel, que es sumamente personalizable y además cuenta con la base idónea (sencilla y sin demasiados complementos). El resto tenía demasiadas funcionalidades que no serían utilizadas en DEFIDNET y tendrían que ser eliminadas de la aplicación final.

- DEFIDNET pretendía ser una aplicación de diseño de IDNs abstracta y alejada de cuestiones técnicas como protocolos de red utilizados, información utilizada en las opciones expuestas con anterioridad.

- El código del sistema inicial estaba escrito en Java, por lo que la integración con los simuladores descritos anteriormente podría resultar complicada. Por su parte, GraphPanel ofrecía una aplicación escrita en Java, lo que permitía un proceso de integración y adición de nuevos módulos y características relativamente sencillo.

Se puede consultar información más detallada sobre GraphPanel en el punto *2.7.2. GraphPanel*.

2.7. Herramientas utilizadas

En esta sección se explican brevemente las herramientas utilizadas en el proyecto DEFIDNET, si bien en el capítulo 6. *Implementación* se puede encontrar información más técnica acerca de éstas y su integración con la aplicación desarrollada.

2.7.1. Java

Java es un lenguaje orientado a objetos. Destaca especialmente por su capacidad de adaptación a distintos dispositivos y sistemas operativos bajo el lema: *"Write one, run everywhere"* [33].

Se seleccionó Java como lenguaje de programación frente a otros que permiten realizar aplicaciones gráficas como C# o Python por una serie de razones:

- El proyecto de partida estaba codificado en Java. Esta es la razón más obvia, puesto que el proceso de integración de las nuevas partes es sencillo y natural. La elección de otro lenguaje de programación hubiese supuesto mucho más tiempo y esfuerzo.
- Es multiplataforma. Desde un inicio se decidió que la aplicación DEFIDNET fuese compatible tanto con Windows como con Linux. Java destaca por su capacidad de adaptación.
- La autora del presente Trabajo de Fin de Grado tenía un buen nivel de conocimiento en Java. Esto significó una mejora en la productividad, puesto que no existía una curva de aprendizaje que entorpeciera el desarrollo en las fases iniciales de codificación.

El desarrollo de la aplicación se apoyó fundamentalmente en el uso de Java Swing para la consecución de la interfaz gráfica. Java Swing es una biblioteca gráfica de

Java que permite diseñar y crear interfaces mediante el uso de distintos componentes como frames, diálogos, botones o listas, los cuales son gestionados por eventos.

Java Swing proporcionaba la funcionalidad perfecta para comenzar la implementación de la aplicación, si bien hubo que añadir componentes externos para suplir las necesidades del proyecto. Dichos elementos externos se exponen en los siguientes puntos.

2.7.2. GraphPanel

GraphPanel[4] es una herramienta desarrollada por el Dr. John B. Matthews sobre la cual se sustenta el visualizador interactivo de DEFIDNET. Dicho visualizador representa la funcionalidad añadida más importante del programa, que consiste en permitir al usuario diseñar IDNs mediante figuras interactivas.

Si bien la funcionalidad ofrecida originalmente es completa y suficiente para el propósito que fue creada, el estado de desarrollo en el que se encontraba era insuficiente para el uso que se le quería dar en la aplicación DEFIDNET. Sin embargo, representaba una muy buena base para conseguir la funcionalidad deseada.

En la figura 2.7 se puede ver una captura del GraphPanel original, sin modificación alguna.

[4]GraphPanel. https://sites.google.com/site/drjohnbmatthews/graphpanel

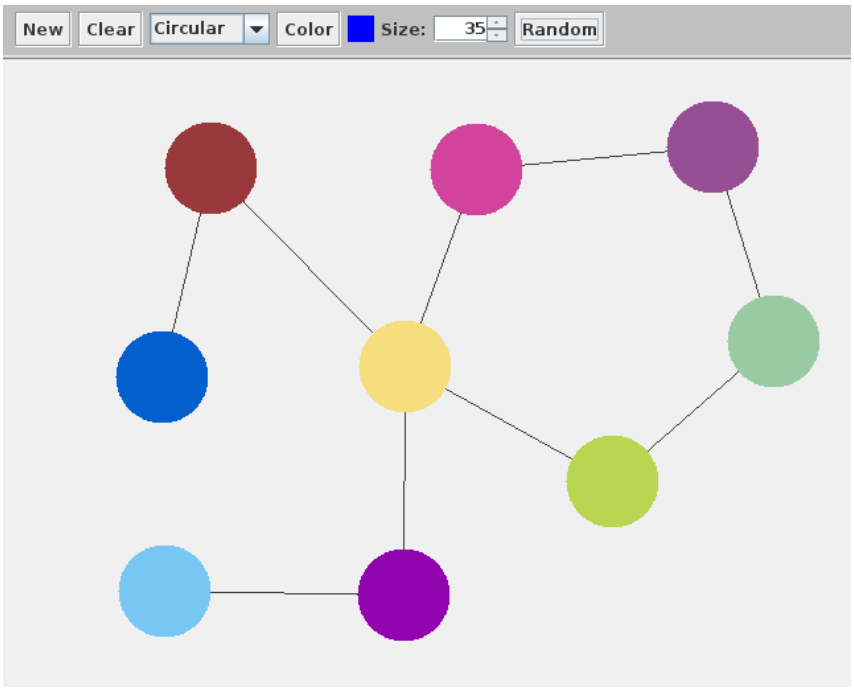

Figura 2.7: Captura de pantalla de GraphPanel.

En la sección *2.6. Situación actual* se puede encontrar el estudio realizado para la elección de la herramienta utilizada en la implementación del visualizador interactivo, parte más importante del presente proyecto y finalmente basada en GraphPanel.

2.7.3. GraphViz

GraphViz[5] es un software de generación de gráficos. Se ha utilizado para la exportación a formato imagen de una IDN y para visualizar IDNs demasiado grandes que no podían ser visualizadas en el visualizador interactivo por motivos de usabilidad y escalabilidad.

[5]GraphViz. http://www.graphviz.org/

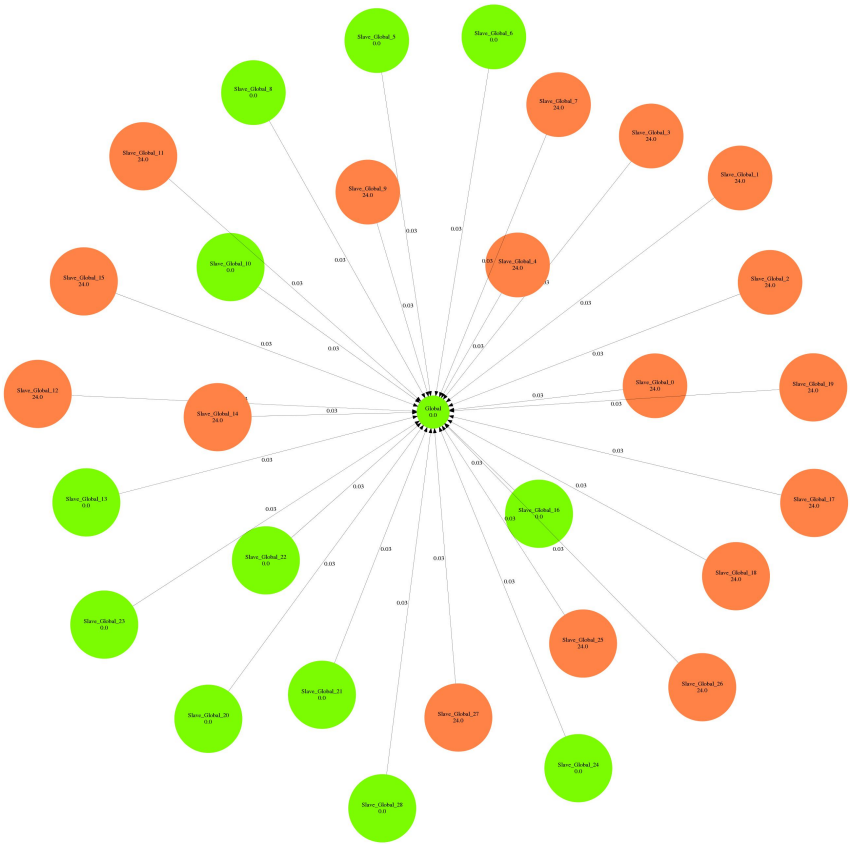

Figura 2.8: Captura de IDN exportada a imagen mediante GraphViz.

Si bien se tuvieron en cuenta otras herramientas similares para la exportación de redes a formato imagen como JUNG [6], JGraphX [7] o JIVE [8], finalmente se optó por GraphViz puesto que el prototipo de DEFIDNET utilizaba este software (aunque de manera externa) y las funcionalidades proporcionadas por las distintas opciones eran muy similares.

[6] JUNG. http://jung.sourceforge.net/
[7] JGraphX. https://github.com/jgraph/jgraphx
[8] JIVE. http://www.cse.buffalo.edu/jive/

2.7.4. ECJ

ECJ es un sistema de investigación EC[9] desarrollado en Java por el laboratorio de computación evolutiva de la universidad George Mason[10]. Es el encargado de ofrecer los algoritmos evolutivos que permiten encontrar una solución óptima coste-beneficio para una IDN mediante búsqueda evolutiva.

Esta herramienta formaba parte de la primera versión de DEFIDNET (prototipo de la aplicación desarrollada en este proyecto), y no requirió de ninguna modificación interna. Únicamente se modificó la forma de pasar los parámetros (antes por fichero) y de recogerlos, utilizando la interacción del usuario con la interfaz gráfica. Por lo tanto, se puede decir que este bloque es tratado a modo de caja negra, introduciendo la información necesaria y recibiendo la salida deseada conociendo sólo a grandes rasgos el modo en el que opera.

No se considera necesario entrar más en detalle sobre las características de ECJ en este apartado, puesto que fue implementado con anterioridad y trata cuestiones complejas de computación evolutiva. Si bien se ha visto necesaria su mención y la explicación de lo arriba expuesto, debido a su relevancia en el conjunto de la aplicación.

En el apartado *6.1.3.8 Generación de soluciones* del capítulo referido a la implementación de DEFIDNET se puede encontrar más información sobre algunas de sus características y sobre los algoritmos utilizados.

2.7.5. JFreeChart

JFreeChart[11] es una librería de Java utilizada para crear de manera sencilla gráficos a partir de una serie de datos proporcionada.

[9] *Evolutionary Computation.*
[10] Página web de George Mason University. http://cs.gmu.edu/~eclab/projects/ecj/
[11] JFreeChart. http://www.jfree.org/jfreechart/

Es una herramienta muy potente que permite generar una gran cantidad de diferentes tipos de gráficos y tablas, así como modificar numerosas características que permiten un grado de personalización muy alto.

Sin embargo, la característica más importante por la cual se decidió su uso frente a otras alternativas como R[12] o Matlab[13] es que permite la interacción del usuario con su interfaz, haciendo posible la selección de coordenadas (posibilitando incluso la corrección automática de esas coordenadas para ajustarse a puntos predefinidos con anterioridad).

Este hecho hizo que se escogiera JFreeChart, debido a la necesidad de permitir al usuario de DEFIDNET interactuar de manera visual con la información devuelta (gráfico de resultados) por la generación de una solución para aplicar contramedidas según el punto escogido.

En la figura 6.13 se puede encontrar un ejemplo de la utilización de JFreeChart en el proyecto, mostrando los puntos obtenidos en el proceso de generación de una solución.

[12]R. http://www.r-project.org/
[13]Matlab. http://www.mathworks.es/products/matlab/

Figura 2.9: Captura de JFreeChart.

Capítulo 3

Gestión del proyecto

3.1. Planificación

3.1.1. Estimación del tiempo

En este apartado se realiza la estimación del tiempo necesario para la finalización del proyecto, desglosando el trabajo total en tareas individuales que permitan definir de manera sencilla y precisa el tiempo requerido para la consecución de cada una de ellas.

El modelo de desarrollo utilizado es el modelo en cascada, como se especifica en la sección *3.2. Metodología*.

Se deben tener en cuenta las dependencias entre las distintas tareas, así como posibles imprevistos que puedan producirse y afectar al tiempo estimado del proyecto global. Por tanto, se debe especificar una duración para cada una de ellas coherente y teniendo en cuenta posibles adversidades.

El objetivo de esta planificación es comparar lo estimado con el tiempo real de desarrollo que ha supuesto el proyecto. De esta manera, se puede ver fácilmente la desviación sufrida entre tiempo planificado y tiempo real y si la metodología

utilizada es correcta y eficaz.

Se han producido variaciones entre la planificación real y el tiempo final empleado en la finalización del proyecto (como se puede consultar en el punto *3.1.1.2 Tiempo real empleado*). Se considera necesario destacar algunas consideraciones a tener en cuenta cuando el lector analice el resultado obtenido:

- El tiempo estimado inicial de desarrollo del proyecto era de 5 meses. Comenzando a finales de febrero de 2014 y finalizando en julio de 2014. Se consideró un tiempo adecuado para la consecución del proyecto, dedicando las horas estipuladas (5 horas/día, 22 días al mes) e incluyendo posibles imprevistos que pudieran surgir durante el proceso.
- El tiempo final empleado en el proyecto ha sido de 7 meses. Se comenzó a principios de marzo de 2014 y se ha finalizado en septiembre de 2014. Ésto se ha producido por cuestiones laborales y académicas de la alumna. Debido a una sobrecarga de trabajo en ambos ámbitos fue imposible seguir el ritmo marcado (*a priori* coherente) para el desarrollo del proyecto y fue necesario tomar 2 meses de suspensión en los cuales ésta priorizó tareas externas al proyecto de cara a finalizar sus estudios. Después de esos meses se volvió al trabajo de manera inmediata y se continuó desarrollando el proyecto siguiendo la planificación marcada, aunque con un desfase de tiempos. Esto, derivó en un retraso obvio de 2 meses que repercutió en los plazos previstos de entrega.

3.1.1.1. Planificación inicial

En la figura 3.1 se expone el diagrama de Gantt realizado durante la fase de planificación del proyecto.

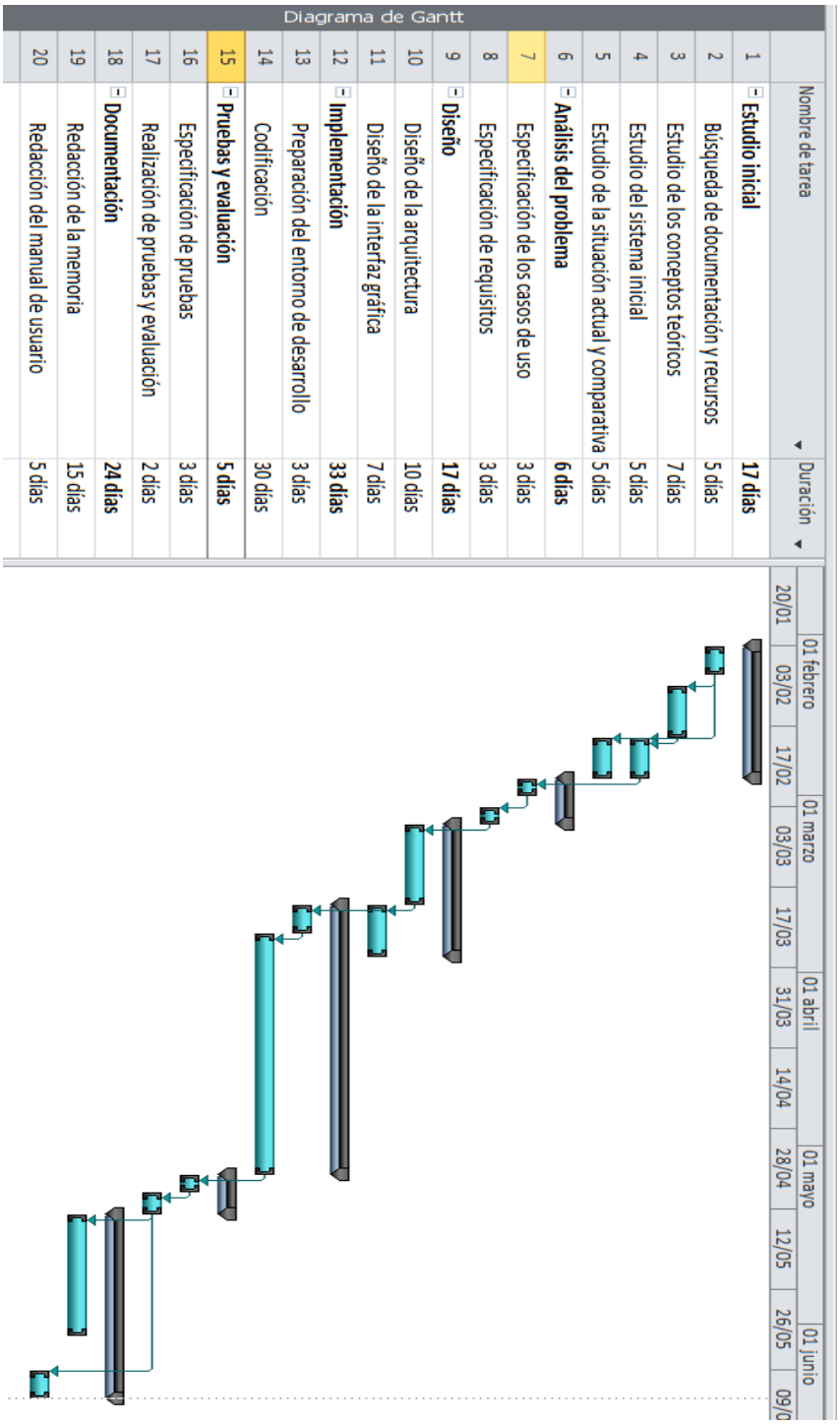

Figura 3.1: Diagrama de Gantt de la planificación inicial.

3.1.1.2. Tiempo real empleado

Debido a las razones expuestas con anterioridad en este apartado de planificación, la estimación inicial no pudo llevarse a cabo.

De los 5 meses inicialmente planificados para el desarrollo del proyecto, finalmente han sido necesarios 7 meses para su completa finalización. Por tanto, se observa un retraso de 2 meses respecto a lo estimado.

Cabe destacar que el retraso producido se debe a razones externas al proyecto, que nada tienen que ver con dificultades en su desarrollo o problemas de planificación de tiempos en las tareas. Salvando ese retraso inicial, la estimación ha resultado bastante exacta para cada tarea individual.

Por tanto, el tiempo real empleado en el desarrollo del proyecto ha sido de **7 meses**.

3.1.2. Coste del proyecto

En el presente apartado se realiza el análisis económico inicial del proyecto analizando los costes que se prevén durante el mismo. Se estiman los costes necesarios para el desarrollo de dicho proyecto diferenciando según su origen:

- **Recursos humanos**. Costes derivados de los recursos humanos utilizados en el desarrollo del proyecto. El coste total del personal requerido se lleva a cabo teniendo en cuenta el salario en base a la categoría a la que se pertenezca, el número de horas trabajadas y la duración del proyecto. Además, se le añade el 23,60 % correspondiente a la seguridad social [25].
- **Hardware**. Costes referidos a la compra de los equipos y periféricos necesarios. Por ejemplo: portátiles, impresoras, escáneres,etc. El coste imputable al proyecto de dicho hardware se calcula en base a la vida útil del equipo y a la duración del proyecto.

- **Software**. Costes procedentes de la compra de los programas y utilidades necesarios para desarrollar el proyecto. De la misma manera que en el punto anterior, se calcula el coste imputable a éste.

- **Bienes fungibles**. Costes de los consumibles, como son el papel, la tinta o el material de oficina.

3.1.2.1. Costes de los recursos humanos

A continuación se muestran los costes derivados del trabajo humano realizado.

Antes de calcular los costes, se deben tener en cuenta las siguientes consideraciones:

- El proyecto cuenta con un equipo de trabajo de un único miembro.

- El salario base se calcula siguiendo la información proporcionada por el *Ministerio de Empleo y Seguridad Social* [25] en el apartado de bases y tipos de cotización para el año 2014.

- El miembro del equipo ha trabajado 7 meses (154 días).

- Se fijan 22 días como laborales cada mes.

- Las horas trabajadas por el equipo de trabajo (compuesto por un único miembro) al día son 5 horas.

En la tabla 3.1 se expone la información extraída del *Ministerio de Empleo y Seguridad Social* sobre el salario base mínimo y máximo para cada categoría profesional.

Grupo de cotización	Categoría profesional	Bases mínimas euros/mes	Bases máximas euros/mes
1	Ingenieros y Licenciados.Personal de alta dirección no incluido en el artículo 1.3.c) del Estatuto de los Trabajadores	1.051,50	3.597,00

Tabla 3.1: Categoría profesional y retribuciones.

Se calcula que la base media en euros/mes es de 2.324,25 euros/mes para un trabajador a tiempo completo (8 horas). Esta cifra se tiene en cuenta para calcular el coste en euros/hora que el trabajador supone para la empresa: 13,20 euros/hora de media (22 días laborables, 8 horas al día).

Se utiliza el dato anterior como coste euro/hora que supone el trabajador para este proyecto en concreto.

Datos aplicados al equipo de trabajo del proyecto:

Apellidos y nombre	Grupo de cotización	Coste euros/hora	Dedicación horas/día	Duración proyecto en días	Coste total en euros
Canes López, Marta	1	13,20	5	154	10.164,00

Tabla 3.2: Información recursos humanos empleados.

Al coste expuesto en la tabla anterior se le debe aplicar el coste derivado de la seguridad social, que para la empresa supone el 23,60 % del sueldo base del trabajador según la tabla 3.3, extraída del *Ministerio de Empleo y Seguridad social.*

Contingencias	Empresa (%)	Trabajadores (%)	Total (%)
Comunes	23,60	4,70	28,30
Horas extraordinarias Fuerza Mayor	12,00	2,00	14,00
Resto horas extraordinarias	23,60	4,70	28,30

Tabla 3.3: Información seguridad social.

Una vez añadido el coste de la seguridad social queda la siguiente información para los 7 meses de proyecto:

Apellidos y nombre	Coste total en euros	Cotización (%)	Coste total con cotización en euros
Canes López, Marta	10.164,00	23,60	12.501,72

Tabla 3.4: Costes recursos humanos.

Por tanto, el coste total concerniente a los recursos humanos empleados durante el desarrollo del proyecto es de **12.501,72 euros**.

3.1.2.2. Costes de hardware

Los costes de hardware sólo son atribuibles a un portátil utilizado para el desarrollo. No ha sido necesario recurrir a la compra de impresoras u otros equipos adicionales. En la tabla 3.5 se puede ver el coste incurrido en el equipamiento hardware.

Hardware	Precio unitario (sin I.V.A) en euros	Vida útil en meses	Uso destinado al proyecto en meses	Coste en euros
Portátil Asus A53J	473,21	36	7	92,01

Tabla 3.5: Costes hardware.

El coste total del hardware utilizado para el proyecto es de **92,01 euros**.

3.1.2.3. Costes de software

A continuación se detallan los costes derivados de la compra del software requerido.

Durante el proyecto, ha sido necesario utilizar dos sistemas operativos para comprobar la funcionalidad de la aplicación en Windows y Linux, puesto que está diseñada para funcionar efectivamente en ambos sistemas.

El coste del sistema operativo *Windows 7 Standard* está incluido en el precio del portátil especificado en el apartado anterior, como parte del software ofrecido por defecto.

Así mismo, es importante remarcar que también se tienen en cuenta (si los hubiera) los costes de las librerías y herramientas software necesarias para ampliar la funcionalidad de la aplicación. Si bien, en este caso todas las herramientas utilizadas son gratuitas o de libre distribución.

En la tabla 3.7 se muestra una relación del software utilizado y su coste.

Software	Precio unitario (sin I.V.A) en euros	Duración de la licencia en meses	Uso destinado al proyecto en meses	Coste total en euros
Fedora 21	0,00	0*	7	0,00
Microsoft Office 365 Home	78,21	12	7	45,62
Microsoft Project 365	194,60**	7	7	194,60
Netbeans IDE 8.0	0,00	0*	7	0,00
GraphViz 2.38	0,00	0*	7	0,00
JFreeChart 1.0.18	0,00	0*	7	0,00
ECJ 21	0,00	0*	7	0,00
Total				240,22

Tabla 3.6: Costes software.

** El valor 0 en la duración de la licencia en meses especificada en la tabla anterior significa que la herramienta o programa es gratuito, y que carece de meses de licencia. Se entiende que su derecho de uso es de por vida.*

*** La suscripción de Microsoft Project 365 (y del resto de productos 365 de Microsoft) se debe ir renovando mes a mes. Por lo tanto, en este proyecto solo se renovó la licencia durante los 7 meses que duró el desarrollo de éste.*

El coste total del software calculado es **240,22 euros**.

3.1.2.4. Costes de bienes fungibles

Se ha establecido los costes provenientes de la provisión de gastos fungibles (papel, tinta, material de oficina, etc.) en **50,00 euros**.

3.1.2.5. Costes totales

En este apartado se aúnan los distintos costes especificados en los puntos anteriores, añadiendo los costes indirectos y el I.V.A:

Origen de los costes		Coste total en euros
Costes recursos humanos		12.501,72
Costes hardware		92,01
Costes software		240,22
Costes bienes fungibles		50,00
Subtotal (TCD)		12.883,95
Costes indirectos (15 % del TCD)	15 %	1.932,59
Impuesto sobre el valor añadido (I.V.A)	21 %	2.705,62
Total		17.522,16

Tabla 3.7: Costes totales del proyecto.

Por tanto, el coste total del proyecto calculado es de **diecisiete mil quinientos veintidós euros y dieciséis céntimos, 17.522,16 euros**.

3.2. Metodología

Para el desarrollo del proyecto se ha utilizado el modelo en cascada. Éste, divide en etapas rigurosamente ordenadas el ciclo de vida del proyecto: hasta que la fase actual no es finalizada no se puede comenzar la siguiente. Esto hace que sea necesario seguir la planificación de manera estricta y ajustarse a los tiempos definidos lo máximo posible, puesto que si se produce un retraso en una de las etapas, el proyecto en su totalidad se ve retrasado.

El modelo en cascada consta de las siguientes fases:

- **Análisis**. Fase destinada al análisis del problema y a la obtención de requisitos.

- **Diseño**. Fase en la cual se realiza el estudio de las necesidades obtenidas a partir de los requisitos y de la funcionalidad necesaria. Se realiza el diseño de componentes y de clases.
- **Codificación**. Fase de implementación del programa, en el cual se desarrolla el software siguiendo las pautas de diseño dispuestas con anterioridad.
- **Pruebas**. En esta fase se comprueba que el programa no tiene errores y cumple los requisitos especificados al inicio del proyecto.

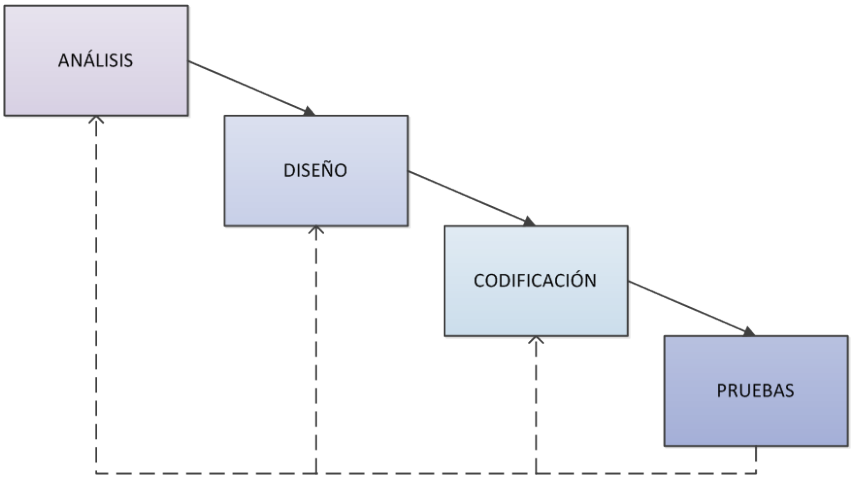

Figura 3.2: Ciclo de vida de un proyecto siguiendo el modelo en cascada.

Durante el transcurso del proyecto, se han mantenido reuniones periódicas entre el tutor y la alumna en las cuales se ha analizado el estado del proyecto y la productividad para verificar que el desarrollo de éste se llevaba a cabo correctamente y en los tiempos adecuados. De la misma manera, se han expuesto los problemas encontrados. Esta práctica permite recibir *feedback*[1] sobre las partes desarrolladas hasta el momento y mejorar de cara a futuros proyectos.

[1]Retroalimentación.

3.3. Control de versiones

En un proyecto de estas dimensiones, la gestión de versiones es un aspecto sumamente importante que permite tener un respaldo de la información y poder acceder al estado del código y demás ficheros en puntos anteriores en el tiempo. Es necesaria la utilización de una herramienta que controle el desarrollo del proyecto de manera eficaz.

Para el control de versiones se ha utilizado Git[2], una sencilla pero potente herramienta que permite realizar un amplio conjunto de acciones sobre el código versionado.

Si bien Git es especialmente útil en entornos colaborativos con equipos de trabajo de varias personas gestionando el mismo código a la vez, también es eficaz utilizado por una única persona de manera local con el objetivo de poder realizar diferentes operaciones sobre el código además de guardar estados anteriores.

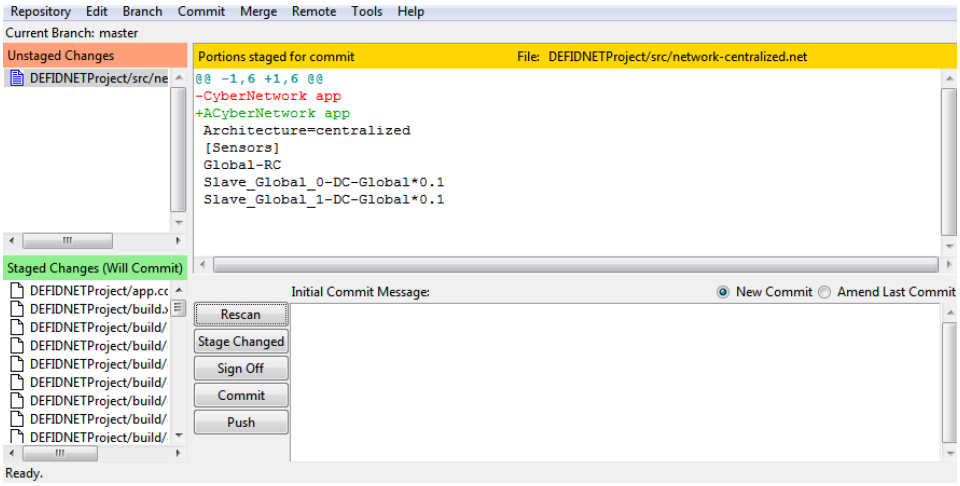

Figura 3.3: Captura de pantalla de Gitk, interfaz gráfica de Git.

[2]Página web de Git. http://git-scm.com/

Capítulo 4

Análisis del problema

En el presente capítulo se analiza el problema a resolver mediante la especificación de casos de uso y requisitos.

De esta manera, se obtiene la información necesaria para diseñar y desarrollar la aplicación DEFIDNET correctamente, habiendo definido previamente los objetivos que se pretenden conseguir de manera clara, así como las secuencias de operaciones que la aplicación debe cumplir.

4.1. Casos de uso

Los casos de uso muestran las secuencias de interacciones entre el usuario de la aplicación (actor) y el sistema, usualmente mediante diagramas o tablas. Así mismo, sirven para refinar un conjunto de requisitos de acuerdo a una función o tarea.

En las siguientes secciones se presentan los casos de uso definidos para cada secuencia de operaciones que ha de llevar a cabo el usuario para utilizar las distintas funcionalidades de DEFIDNET.

4.1.1. Diagramas de casos de uso

Los diagramas expuestos en las figuras 4.1 y 4.2 describen de manera gráfica las distintas operaciones que un usuario puede realizar dentro del ámbito de la aplicación. En el punto *4.1.2. Tablas de casos de uso* se puede encontrar información más detallada acerca de las precondiciones, secuencia de pasos y demás características asociadas a una operación determinada.

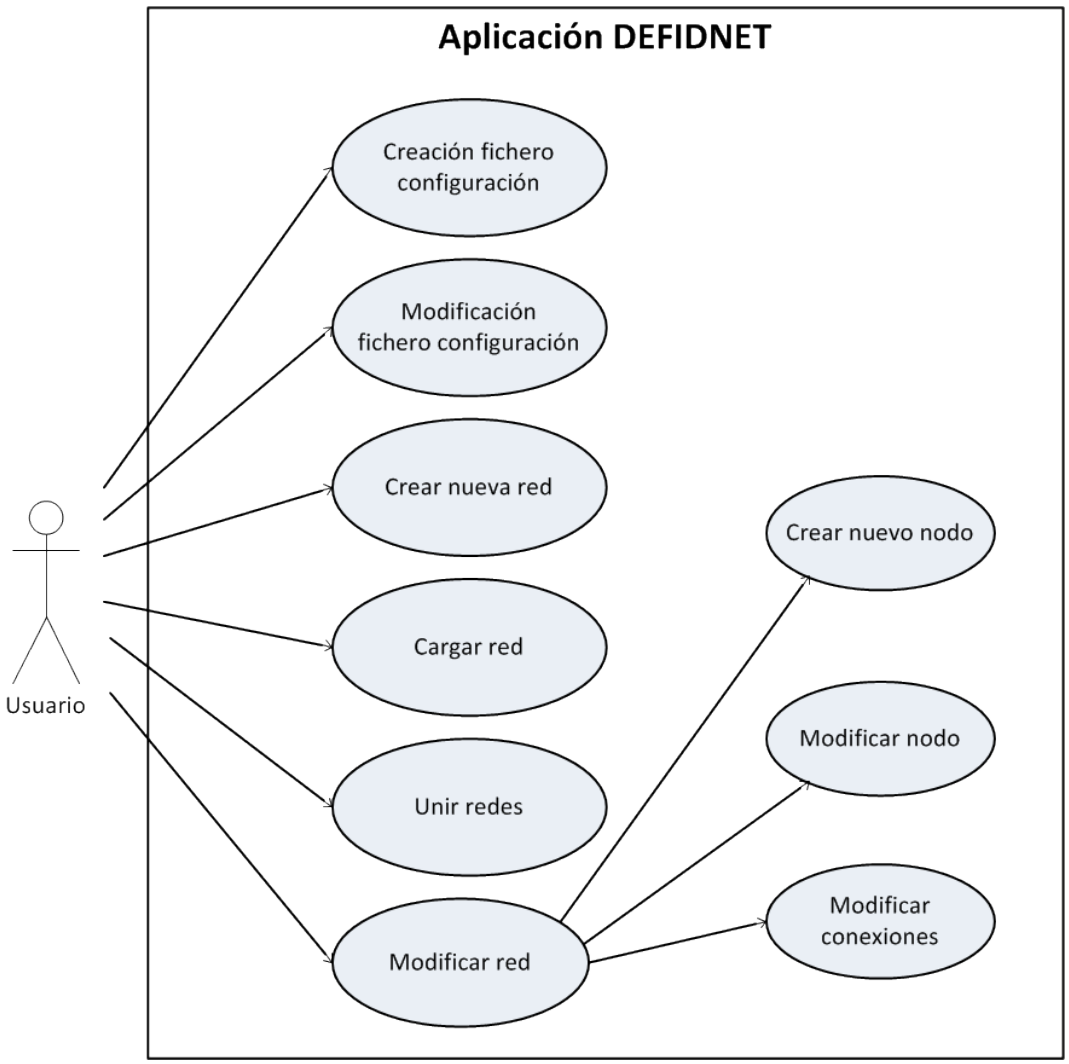

Figura 4.1: Diagrama de casos de uso I.

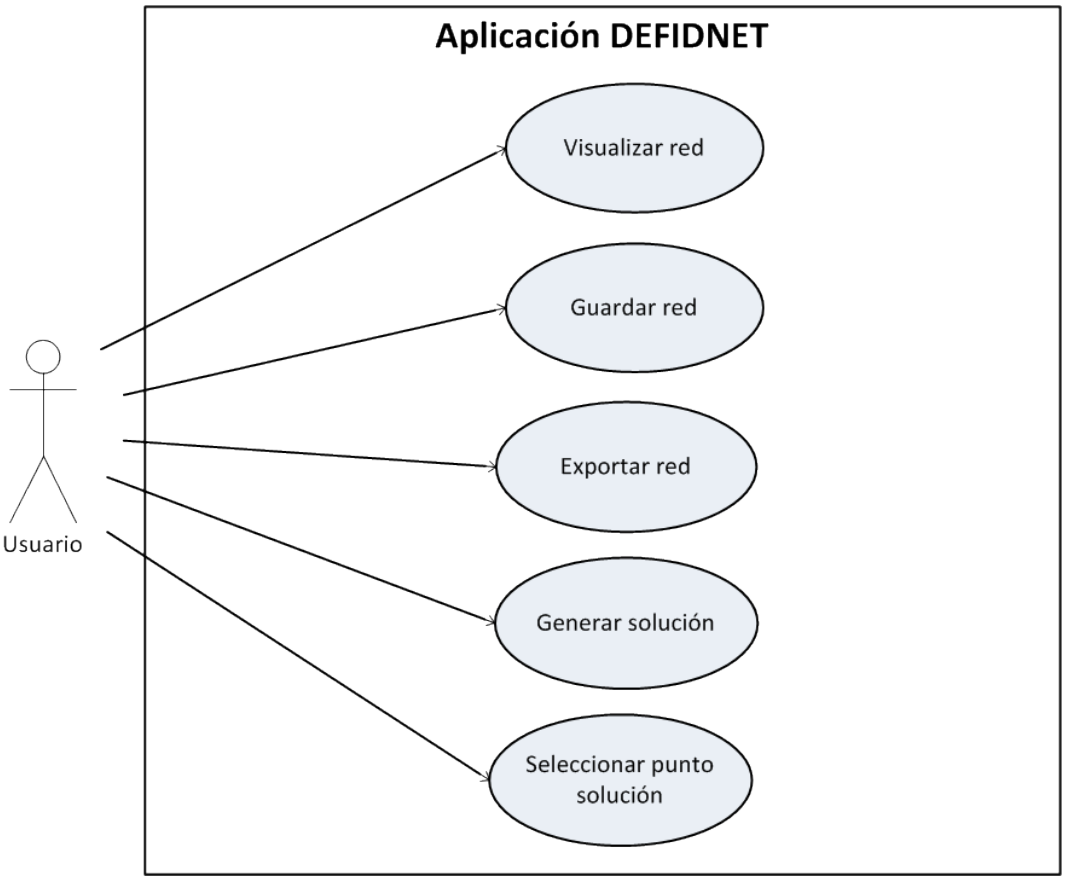

Figura 4.2: Diagrama de casos de uso II.

4.1.2. Tablas de casos de uso

En este punto se describen de manera más específica los casos de uso definidos, añadiendo información adicional a los diagramas expuestos arriba.

El formato de las tablas para la definición de los casos de uso es la siguiente:

- **ID Caso de uso**. Identificador. Sigue el formato CU-XXX, donde XXX indica el número de caso de uso.
- **Actores**. Roles de los usuarios respecto al sistema. En este caso, solo existe

un tipo de actor, que es el usuario de la aplicación.

- **Título**. Título breve y significativo que permite identificar fácilmente la funcionalidad a la que hace referencia el caso de uso.
- **Descripción**. Descripción más extensa que explica de manera más detallada el caso de uso y sus características.
- **Precondiciones**.Condiciones previas que se deben cumplir para poder iniciar el caso de uso.
- **Postcondiciones**. Estado posterior al caso de uso de la aplicación.
- **Secuencia de pasos**. Listado de acciones necesarias para llevar a cabo correctamente un caso de uso.
- **Excepciones**. Posibles situaciones alternativas que pueden darse durante la ejecución de la secuencia de pasos.

A continuación, pasan a describirse los distintos casos de uso especificados.

ID Caso de uso	CU-001	
Título	Crear fichero de configuración	
Actores	Usuario	
Descripción	El usuario rellena el formulario para la creación del fichero de configuración inicial si éste no es encontrado por la aplicación al inicio.	
Precondiciones	El usuario ha iniciado la aplicación.	
Postcondiciones	El fichero de configuración ha sido creado correctamente.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El sistema verifica si existe el fichero de configuración.
	3	El usuario rellena el formulario para la creación del fichero de configuración inicial.
Excepciones	Paso	Acción
	2	El fichero de configuración puede encontrarse ya creado.

Tabla 4.1: Caso de uso CU-001.

ID Caso de uso	CU-002	
Título	Modificar fichero de configuración	
Actores	Usuario	
Descripción	El usuario rellena el formulario para la modificación del fichero de configuración para editar su información.	
Precondiciones	El usuario ha iniciado la aplicación.	
Postcondiciones	El fichero de configuración ha sido modificado correctamente.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario selecciona la opción *GraphViz Settings*, cuyo botón tiene la figura de ruedas dentadas.
	3	El usuario rellena el formulario para la modificación del fichero de configuración inicial.
Excepciones	Paso	Acción
	2	El fichero de configuración puede no existir o no ser encontrado por la aplicación.

Tabla 4.2: Caso de uso CU-002.

ID Caso de uso	CU-003	
Título	Crear nueva IDN	
Actores	Usuario	
Descripción	El usuario introduce la información pedida en los formularios de creación de una IDN para crear una nueva.	
Precondiciones	El usuario ha iniciado la aplicación.	
Postcondiciones	La IDN ha sido creada correctamente y es visualizada por el usuario.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario selecciona la opción *New IDN* del menú principal.
	3	El usuario rellena el formulario general para la creación de una IDN.
	4	El usuario rellena el formulario específico para cada posible rol de los nodos de la IDN.
Excepciones	Paso	Acción
	1	El fichero de configuración inicial puede no ser encontrado. El usuario deberá crearlo (Caso de uso C-001).
	3	Los datos introducidos pueden ser no válidos. El usuario visualizará una ventana de error.
	4	Los datos introducidos pueden ser no válidos. El usuario visualizará una ventana de error.

Tabla 4.3: Caso de uso CU-003.

ID Caso de uso	CU-004	
Título	Cargar IDN ya existente	
Actores	Usuario	
Descripción	El usuario selecciona un fichero que almacena una IDN previamente guardada.	
Precondiciones	El usuario ha iniciado la aplicación.	
Postcondiciones	La IDN ha sido cargada y es visualizada por el usuario.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario selecciona la opción *Load IDN* del menú principal.
	3	El usuario selecciona un fichero de extensión .net mediante el gestor de ficheros.
Excepciones	Paso	Acción
	1	El fichero de configuración inicial puede no ser encontrado. El usuario deberá crearlo (Caso de uso C-001).
	3	El fichero seleccionado por el usuario puede estar corrupto o no tener el formato exigido por la aplicación. La aplicación mostrará un mensaje de error y el usuario deberá seleccionar otro fichero .net.
	3	El usuario puede seleccionar un fichero de una extensión distinta a la de .net. La aplicación mostrará un mensaje de error y el usuario deberá seleccionar un fichero .net.

Tabla 4.4: Caso de uso CU-004.

ID Caso de uso	CU-005
Título	Unir IDNs
Actores	Usuario
Descripción	El usuario puede combinar distintas IDNs en una única (IDN de arquitectura *joined*) seleccionando los nodos que actuarán como uniones para cada una.
Precondiciones	El usuario ha iniciado la aplicación.
Postcondiciones	La nueva IDN conjunta ha sido creada y es visualizada por el usuario.

Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario selecciona la opción *Join IDNs* del menú principal.
	3	El usuario selecciona mediante un gestor de ficheros una o más IDNs que desea unir.
	4	El usuario selecciona en un formulario los nodos que actuarán como uniones.
Excepciones	**Paso**	**Acción**
	1	El fichero de configuración inicial puede no ser encontrado. El usuario deberá crearlo (Caso de uso C-001).
	3	El usuario puede seleccionar IDNs en formato `.net` que no pueden ser leídas o estar corruptas. La aplicación mostrará una mensaje de error.
	3	El usuario puede seleccionar un fichero con un formato distinto de `.net`. La aplicación mostrará un mensaje de error.
	3	El usuario puede no seleccionar ningún nodo. La aplicación mostrará un mensaje de error.

Tabla 4.5: Caso de uso CU-005.

ID Caso de uso	CU-006	
Título	Crear nuevo nodo	
Actores	Usuario	
Descripción	El usuario crea un nuevo nodo en la IDN actual	
Precondiciones	El usuario ha creado una IDN nueva (CU-003) o ha cargado una ya existente (CU-004). La IDN debe ser de un tamaño inferior a 50 nodos y tener una arquitectura distinta de la *joined*.	
Postcondiciones	El nuevo nodo ha sido añadido a la IDN y el usuario visualiza la IDN actualizada.	
Secuencia	Paso	Acción
	1	El usuario selecciona la opción *New node* del menú superior del visualizador interactivo de la IDN.
	2	El usuario rellena el formulario de creación de nuevo nodo.
Excepciones	Paso	Acción
	2	El usuario puede introducir valores no válidos en el formulario. La aplicación mostrará un mensaje de error y el usuario deberá introducir de nuevo los valores.

Tabla 4.6: Caso de uso CU-006.

ID Caso de uso	CU-007	
Título	Modificar nodo	
Actores	Usuario	
Descripción	El usuario modifica un nodo ya existente en la IDN actual.	
Precondiciones	El usuario ha creado una IDN nueva (CU-003) o ha cargado una ya existente (CU-004). La IDN debe ser de un tamaño inferior a 50 nodos y tener una arquitectura distinta de la *joined*.	
Postcondiciones	El nodo modificado ha sido actualizado en la IDN y el usuario visualiza la nueva IDN.	
Secuencia	Paso	Acción
	1	El usuario selecciona la opción *Edit* del menú flotante asociado a cada nodo de la IDN.
	2	El usuario rellena el formulario de modificación de un nodo.
Excepciones	Paso	Acción
	2	El usuario puede introducir valores no válidos en el formulario. La aplicación mostrará un mensaje de error y el usuario deberá introducir de nuevo los valores.

Tabla 4.7: Caso de uso CU-007.

ID Caso de uso	CU-008	
Título	Eliminar nodo	
Actores	Usuario	
Descripción	El usuario elimina un nodo existente en la IDN actual.	
Precondiciones	El usuario ha creado una IDN nueva (CU-003) o ha cargado una ya existente (CU-004). La IDN debe ser de un tamaño inferior a 50 nodos y tener una arquitectura distinta de la *joined*.	
Postcondiciones	El nodo eliminado ha sido borrado de la IDN y todas sus conexiones eliminadas.	
Secuencia	Paso	Acción
	1	El usuario selecciona la opción *Delete* del menú flotante asociado a cada nodo de la IDN.
Excepciones	Paso	Acción

Tabla 4.8: Caso de uso CU-008.

ID Caso de uso	CU-009
Título	Modificar conexiones
Actores	Usuario
Descripción	El usuario edita las conexiones entre nodos (creando, borrando o editando el valor de influencia de éstas).
Precondiciones	El usuario ha creado una IDN nueva (CU-003) o ha cargado una ya existente (CU-004). La IDN debe ser de un tamaño inferior a 50 nodos y tener una arquitectura distinta de la *joined*.
Postcondiciones	La conexión ha sido actualizada y la nueva IDN es visualizada por el usuario.

Secuencia	Paso	Acción
	1	El usuario selecciona dos nodos de la IDN.
	2	El usuario selecciona la opción *Connect* para conectar los dos nodos, *Disconnect* para desconectarlos o *Edit connection* para editar la influencia de la conexión.
	3	El usuario rellena el formulario de modificación de la influencia en el caso de que haya seleccionado *Edit connection*.
Excepciones	**Paso**	**Acción**
	2	El usuario puede haber seleccionado un único nodo. DEFIDNET mostrará un aviso.
	2	El usuario puede intentar conectar dos nodos ya conectados o eliminar/modificar una conexión entre dos nodos que no están unidos. DEFIDNET mostrará un aviso.
	2	El usuario puede introducir valores no válidos en el formulario de modificación. La aplicación mostrará un mensaje de error.

Tabla 4.9: Caso de uso CU-009.

ID Caso de uso	CU-010	
Título	Visualizar IDN	
Actores	Usuario	
Descripción	El usuario visualiza la IDN actual a través del visualizador interactivo de IDNs o del visualizador estático.	
Precondiciones	El usuario ha creado una IDN nueva (CU-003) o ha cargado una ya existente (CU-004).	
Postcondiciones	La IDN actual ha sido visualizada correctamente por el usuario.	
Secuencia	Paso	Acción
	1	La aplicación comprueba si la IDN tiene más de 50 nodos o tiene arquitectura *joined*. Si se cumple alguna de estas características, la IDN se mostrará mediante le visualizador estático. Sino, se utilizará el visualizador interactivo de IDNs.
	2	La aplicación muestra la IDN al usuario según el método correspondiente.
Excepciones	Paso	Acción

Tabla 4.10: Caso de uso CU-010.

ID Caso de uso	CU-011	
Título	Guardar IDN	
Actores	Usuario	
Descripción	El usuario guarda la IDN actual en formato .net.	
Precondiciones	El usuario ha creado una IDN nueva (CU-003) o ha cargado una ya existente (CU-004).	
Postcondiciones	La IDN ha sido guardada correctamente en formato .net.	
Secuencia	Paso	Acción
	1	El usuario selecciona la opción *Save* del menú superior del visualizador de IDNs (tanto el estático como el interactivo).
	2	El usuario selecciona el directorio en el que desea guardar la IDN e introduce un nombre para ésta mediante el gestor de ficheros.
Excepciones	Paso	Acción
	2	El usuario puede intentar sobrescribir un fichero ya existente. La aplicación mostrará un mensaje de aviso y pedirá al usuario su confirmación.

Tabla 4.11: Caso de uso CU-011.

ID Caso de uso	CU-012	
Título	Exportar IDN	
Actores	Usuario	
Descripción	El usuario exporta la IDN actual como una imagen en formato .gif.	
Precondiciones	El usuario ha creado una IDN nueva (CU-003) o ha cargado una ya existente (CU-004).	
Postcondiciones	La IDN ha sido exportada a imagen correctamente en formato .gif.	
Secuencia	Paso	Acción
	1	El usuario selecciona la opción *Export to image* del menú superior del visualizador de IDNs (tanto el estático como el interactivo).
	2	El usuario selecciona el directorio en el que desea guardar la imagen e introduce un nombre para ésta mediante el gestor de ficheros.
Excepciones	Paso	Acción
	2	El usuario puede intentar sobrescribir un fichero ya existente. La aplicación mostrará un mensaje de aviso y pedirá al usuario su confirmación.

Tabla 4.12: Caso de uso CU-012.

ID Caso de uso	CU-013	
Título	Generar solución	
Actores	Usuario	
Descripción	El usuario realiza una petición a la aplicación para la generación de una solución de protección óptima en términos de coste y reducción de riesgo ante ataques.	
Precondiciones	El usuario ha creado una IDN nueva (CU-003) o ha cargado una ya existente (CU-004).	
Postcondiciones	La aplicación muestra un gráfico con los resultados.	
Secuencia	Paso	Acción
	1	El usuario selecciona la opción *Generate Solution* del menú superior del Editor visual de IDNs.
Excepciones	Paso	Acción

Tabla 4.13: Caso de uso CU-013.

ID Caso de uso	CU-014	
Título	Seleccionar punto solución	
Actores	Usuario	
Descripción	Dada una solución de protección, el usuario selecciona un punto concreto para aplicar contramedidas y asegurar la IDN.	
Precondiciones	El usuario ha generado una solución para la IDN.	
Postcondiciones	Se ha asegurado y guardado en un fichero .net correctamente la IDN actual con las contramedidas aplicadas, que produce una reducción del riesgo.	
Secuencia	Paso	Acción
	1	El usuario selecciona la opción un punto del gráfico resultante de la opción *Generate Solution* (CU-011).
Excepciones	Paso	Acción

Tabla 4.14: Caso de uso CU-014.

4.2. Requisitos

A continuación se definen los requisitos del proyecto. Su descripción y especificación es una fase fundamental en el desarrollo de un proyecto de estas características, puesto que sirven para definir los objetivos que debe cumplir el sistema para satisfacer las necesidades especificadas.

Los objetivos principales de la Especificación de Requisitos del Sistema (ERS) son servir como medio de comunicación entre clientes y desarrolladores, y recoger tanto las necesidades de dichos clientes como los requisitos que debe cumplir el sistema software a desarrollar para satisfacer esas necesidades [22].

Esta especificación de requisitos también se compone de los casos de uso del apartado anterior, que suponen la descripción de los requisitos funcionales del sistema.

Los distintos requisitos obtenidos son clasificados en las siguientes categorías:

- **Requisitos de usuario.** Son los requisitos que especifican las necesidades del cliente. Se han extraído a partir de el análisis de la funcionalidad deseada por éste.
- **Requisitos de software.** Son los requisitos que debe cumplir el sistema. Responden a cuestiones más técnicas que el cliente no especifica en las reuniones para la captura de requisitos.

Las tablas de requisitos que se exponen en los siguientes puntos siguen la estructura:

- **ID Requisito**. Identificador. Sigue los formatos RU-XXX (en caso de que el requisito sea un requisito de usuario) y RS-XXX (cuando se trata de un requisito de software), donde XXX indica el número del requisito en su categoría.
- **Título**. Título breve y significativo que permite identificar fácilmente la necesidad y por tanto funcionalidad requerida a la que hace referencia el requisito.
- **Descripción**. Descripción más extensa que explica de manera más detallada el requisito y sus características.

4.2.0.1. Requisitos de usuario

Los requisitos de usuario permiten una descripción clara y concisa de las necesidades que el cliente pretende cubrir con el sistema desarrollado.

ID Requisito	RU-001
Título	Creación de una nueva IDN.
Descripción	La aplicación dispondrá de una opción que permita la generación de una nueva IDN introduciendo los parámetros deseados.

Tabla 4.15: Requisito de usuario RU-001.

ID Requisito	RU-002
Título	Carga de una IDN.
Descripción	La aplicación dispondrá de una opción que permita cargar IDNs ya creadas y almacenadas en formato .net (formato de IDN de la aplicación).

Tabla 4.16: Requisito de usuario RU-002.

ID Requisito	RU-003
Título	Unión de IDNs.
Descripción	La aplicación será capaz de unir dos o más IDNs almacenadas en formato .net mediante la unión de los nodos seleccionados por el usuario.

Tabla 4.17: Requisito de usuario RU-003.

ID Requisito	RU-004
Título	Visualización y edición de una IDN.
Descripción	La aplicación permitirá visualizar las distintas IDNs según su arquitectura y modificar los distintos nodos y conexiones que las componen (no en todos los casos).

Tabla 4.18: Requisito de usuario RU-004.

ID Requisito	RU-005
Título	Guardado de una IDN.
Descripción	La aplicación dispondrá de una opción que permita guardar la IDN actual en un fichero de sistema (formato .net) que almacenará la información correspondiente a dicha IDN y será recuperable.

Tabla 4.19: Requisito de usuario RU-005.

ID Requisito	RU-006
Título	Exportación de una IDN.
Descripción	La aplicación dispondrá de una opción que permita la exportación de la IDN a formato .gif.

Tabla 4.20: Requisito de usuario RU-006.

ID Requisito	RU-007
Título	Inserción de un nuevo nodo en la IDN.
Descripción	La aplicación dispondrá de una opción que permita la creación y adición de un nuevo nodo a la IDN, especificando los parámetros requeridos.

Tabla 4.21: Requisito de usuario RU-007.

ID Requisito	RU-008
Título	Modificación de un nodo de una IDN.
Descripción	La aplicación dispondrá de una opción que permita la modificación de nodos ya existentes, modificando los parámetros deseados dentro de las restricciones existentes para dicho nodo.

Tabla 4.22: Requisito de usuario RU-008.

ID Requisito	RU-009
Título	Eliminación un nodo de una IDN.
Descripción	La aplicación dispondrá de una opción que permita el borrado de un nodo de la IDN.

Tabla 4.23: Requisito de usuario RU-009.

ID Requisito	RU-010
Título	Creación, modificación y eliminación de conexiones entre nodos de una IDN.
Descripción	La aplicación permitirá modificaciones en las conexiones entre nodos seleccionando los dos nodos extremo de cada conexión.

Tabla 4.24: Requisito de usuario RU-010.

ID Requisito	RU-011
Título	Generación de la solución.
Descripción	La aplicación dispondrá de una opción que permita la generación de una solución óptima para la IDN actual en base al criterio coste-beneficio.

Tabla 4.25: Requisito de usuario RU-011.

ID Requisito	RU-012
Título	Visualización de resultados de la generación de una solución.
Descripción	La aplicación mostrará la información derivada de la generación de una solución óptima mediante un gráfico interactivo que muestre el Frente de Pareto de valores óptimos en términos de coste y riesgo mitigado.

Tabla 4.26: Requisito de usuario RU-012.

ID Requisito	RU-013
Título	Aplicación de contramedidas.
Descripción	La aplicación permitirá al usuario interactuar con el gráfico resultante especificado en el requisito anterior, pudiendo éste seleccionar un punto del Frente de Pareto correspondiente a la solución generada. La aplicación utilizará la información recibida para aplicar contramedidas en la IDN en base al coste-beneficio especificado.

Tabla 4.27: Requisito de usuario RU-013.

ID Requisito	RU-014
Título	Definición de código colores para medir riesgo de los nodos de una IDN.
Descripción	La aplicación seguirá un código de colores predefinido para colorear las figuras que representan los distintos nodos de la IDN en función de la probabilidad de ser atacados.

Tabla 4.28: Requisito de usuario RU-014.

ID Requisito	RU-015
Título	Zoom en el visualizador de IDNs.
Descripción	La aplicación permitirá acercar y alejar el gráfico mediante la actualización del grafo de figuras.

Tabla 4.29: Requisito de usuario RU-015.

ID Requisito	RU-016
Título	Información y ayuda en la aplicación.
Descripción	La aplicación dispondrá de elementos de ayuda que faciliten su uso, así como de un manual de usuario que contenga la información básica.

Tabla 4.30: Requisito de usuario RU-016.

ID Requisito	RU-017
Título	Idioma inglés como lenguaje de la aplicación.
Descripción	La interfaz de la aplicación sólo será mostrada en idioma inglés. No se contempla la utilización de otros idiomas por el momento.

Tabla 4.31: Requisito de usuario RU-017.

ID Requisito	RU-018
Título	Aplicación eficiente y rápida.
Descripción	La aplicación deberá finalizar siempre en un tiempo de ejecución razonable, si bien este tiempo dependerá del tamaño de las redes manejadas y la dificultad de encontrar una solución óptima.

Tabla 4.32: Requisito de usuario RU-018.

4.2.0.2. Requisitos de software

Los requisitos de software especifican aspectos técnicos de la aplicación, así como formas de llevar a cabo determinados requisitos de usuario.

ID Requisito	RS-001
Título	GraphViz para la exportación de imágenes.
Descripción	La aplicación utilizará la herramienta GraphViz para la exportación a formato .gif de la IDN actual.

Tabla 4.33: Requisito de software RS-001.

ID Requisito	RS-002
Título	ECJ para la generación de la solución óptima.
Descripción	La aplicación utilizará la herramienta ECJ para realizar el proceso de búsqueda multiobjetivo de la IDN y conseguir la solución óptima coste-beneficio.

Tabla 4.34: Requisito de software RS-002.

ID Requisito	RS-003
Título	JFreeChart para la visualización de resultados de una solución.
Descripción	La aplicación utilizará la herramienta JFreeChart para mostrar los gráficos con la información obtenida de la solución óptima coste-beneficio alcanzada.

Tabla 4.35: Requisito de software RS-003.

ID Requisito	RS-004
Título	Aplicación compatible con Linux y Windows.
Descripción	La aplicación DEFIDNET deberá ser compatible con sistemas operativos Linux y con Windows, siendo totalmente operativa en ambos.

Tabla 4.36: Requisito de software RS-004.

ID Requisito	RS-005
Título	Librerías Java Swing y Java AWT de Java para la implementación de la interfaz gráfica de la aplicación.
Descripción	La interfaz gráfica de DEFIDNET se implementará mediante Java Swing y Java AWT.

Tabla 4.37: Requisito de software RS-005.

ID Requisito	RS-006
Título	Uso de un fichero externo para leer los parámetros de configuración de la herramienta GraphViz.
Descripción	La aplicación recuperará datos relevantes de la herramienta GraphViz al inicio desde un fichero de configuración.

Tabla 4.38: Requisito de software RS-006.

ID Requisito	RS-007
Título	Formulario de generación del fichero de configuración de GraphViz.
Descripción	La aplicación, en caso de no encontrar el fichero de configuración, mostrará un formulario en el cual el usuario deberá especificar la información necesaria.

Tabla 4.39: Requisito de software RS-007.

ID Requisito	RS-008
Título	Formulario de modificación del fichero de configuración de GraphViz.
Descripción	La aplicación dispondrá de un formulario de modificación del fichero de configuración de GraphViz para editar su información.

Tabla 4.40: Requisito de software RS-008.

ID Requisito	RS-009
Título	Visualizador interactivo para visualizar/operar con IDNs de pequeño tamaño.
Descripción	La aplicación utilizará el visualizador interactivo para visualizar y modificar mediante distintas operaciones IDNs de un tamaño inferior a 50 nodos y de arquitectura distinta a la arquitectura *joined*.

Tabla 4.41: Requisito de software RS-009.

ID Requisito	RS-010
Título	Visualizador estático mediante GraphViz para visualizar IDNs de gran tamaño.
Descripción	La aplicación exportará a formato imagen (.gif) mediante GraphViz las IDNs de más de 50 nodos o *joined*, permitiendo su visualización y sólo algunos determinados tipos de operaciones.

Tabla 4.42: Requisito de software RS-010.

Capítulo 5

Diseño

5.1. Arquitectura

En esta sección se explica la arquitectura elegida para el desarrollo de la aplicación DEFIDNET, así como las razones aportadas para su elección.

Se ha utilizado el modelo MVC (*Modelo-Vista-Controlador*) para definir la arquitectura de la aplicación. El modelo MVC hace una distinción entre la interfaz gráfica y la funcionalidad interna del programa, por lo que se consigue una separación entre la presentación y la lógica de negocio.

Este patrón consta de tres partes diferenciadas:

- **Modelo.** El modelo es la parte que opera con los datos, la que implementa la lógica de negocio.
- **Vista.** La vista representa la interfaz gráfica de la aplicación; es decir, los elementos visuales que la forman. Es la parte que el usuario visualiza y a través de la cual interactúa con el sistema.
- **Controlador.** El controlador actúa como un intermediario entre la vista y el modelo, y maneja el flujo de información. Es el encargado de responder a

los eventos ocurridos (normalmente producidos por el usuario de la aplicación y en algunos casos lanzados de manera automática) y de enviar peticiones al modelo, el cual le devuelve una respuesta. Después, el controlador se encarga de modificar la vista para actualizar la información.

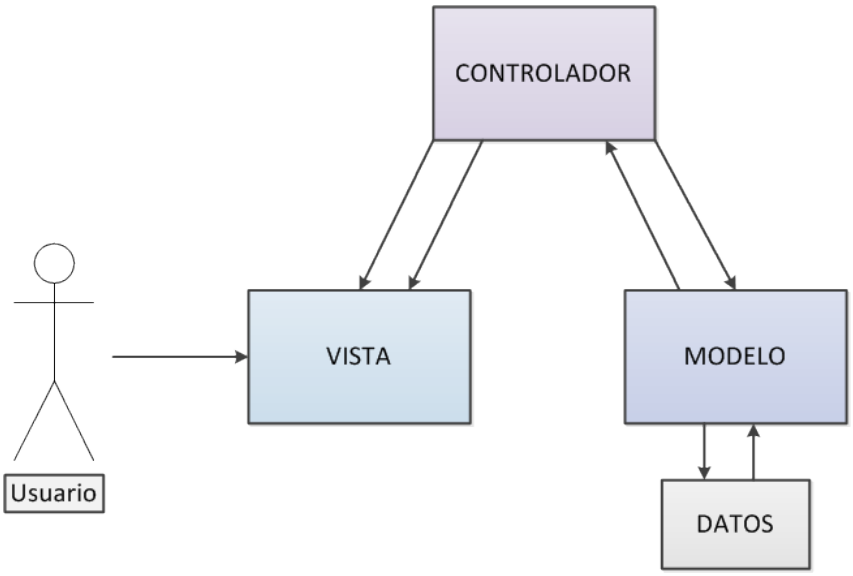

Figura 5.1: Arquitectura *Modelo-Vista-Controlador.*

Se decidió la utilización de este modelo por diversas razones. En primer lugar, proporciona una manera limpia y ordenada de mantener las distintas funcionalidades y módulos separados en secciones lógicas y coherentes. En segundo lugar, realiza una separación rígida de los elementos, que fomenta el seguimiento de los estándares marcados. Y en tercer lugar, porque la alumna (escritora de esta memoria) estaba familiarizada con sus características y forma de implementación, habiendo trabajado con él en varios proyectos.

Para implementar la aplicación DEFIDNET según el diseño especificado se definió el siguiente esquema:

- En la parte del **modelo** se implementarían las funciones para realizar las siguientes operaciones:

- **Definición de IDNs.** Por definición se entiende la creación, modificación, adición de nodos y demás operaciones que cambian las características de la IDN.
- **Operaciones sobre IDNs.** Engloba las operaciones de guardado, carga y exportación a imagen de la IDN, y combinación de varias IDNs en una.
- **Búsqueda de soluciones óptimas.** Funciones que permiten la evolución de la IDN y la consecución de soluciones óptimas coste-beneficio.
- **Gestión de ficheros.** Operaciones de acceso, creación, eliminación y edición de los diferentes ficheros de sistema.

- En la parte de la **vista** se requería la implementación de los siguientes elementos:
  - **Visualizador interactivo de IDNs (de tamaño inferior o igual a 50 nodos).** Permite la visualización y modificación de la IDN con la que el usuario trabaja mediante un entorno interactivo, utilizando grafos de figuras que la representan.
  - **Visualizador estático para IDNs grandes (superiores a 50 nodos).** El visualizador estático es utilizado para visualizar las IDNs exportadas como imágenes en formato .gif, que por su tamaño no pueden ser visualizadas en el visualizador interactivo de IDNs.
  - **Gráfico de resultados de la búsqueda de solución óptima.** En el gráfico de resultados se muestran los resultados obtenidos por la búsqueda de soluciones óptimas. El gráfico es interactivo y permite al usuario seleccionar el punto coste-beneficio deseado.
  - **Interfaz gráfica propiamente dicha de la aplicación.** Se compone de los menús, ventanas, formularios y demás elementos visuales que permiten al usuario visualizar e interactuar con DEFIDNET.

- En la parte del **controlador** se implementaría la siguiente funcionalidad:
    - **Gestión de eventos.** El controlador gestiona los eventos producidos durante la ejecución de la aplicación desencadenados por el usuario o automáticamente.

Se especificó el valor 50 para el número de nodos como valor diferenciador entre IDNs pequeñas y de gran tamaño. Este valor se fijó puesto que se consideró que IDNs demasiado grandes podrían afectar al rendimiento del visualizador interactivo. Así mismo, se llegó a la conclusión de que sería difícil gestionar gráficamente IDNs tan grandes por parte de un administrador y que la visualización final no sería la adecuada, puesto que el panel en el que se dibujan las mismas tiene un tamaño determinado (el máximo posible para cada pantalla) y no sería posible introducir una gran cantidad de nodos en él y que las leyendas fuesen legibles y los nodos y conexiones pudiesen diferenciarse.

Por lo tanto, se optó por la exportación de IDNs superiores a 50 nodos a formato imagen mediante una herramienta externa que pudiese dibujar de manera sencilla y rápida IDNs de grandes dimensiones. En el apartado *2.7. Herramientas utilizadas* se puede encontrar más información sobre GraphViz, herramienta finalmente utilizada. Igualmente, el usuario tiene la opción de modificar directamente las características de la IDN mediante la edición manual ficheros de texto (`.net`).

En la figura 5.2 se expone un diagrama del esquema descrito arriba.

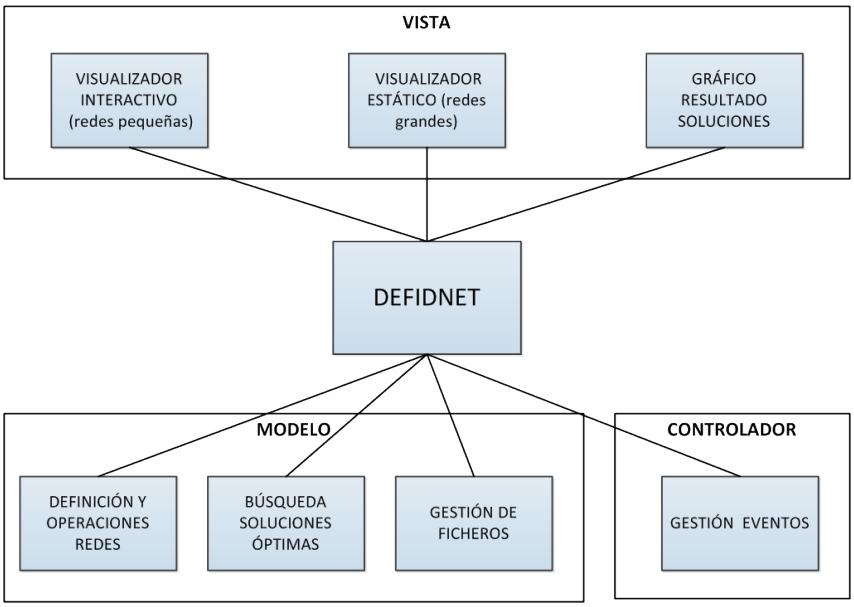

Figura 5.2: Diagrama del diseño general de la aplicación.

Se puede ver que cada capa consta de distintos módulos que conforman DEFIDNET, cada uno encargado de proporcionar funcionalidad vital a la aplicación. Estos módulos interaccionan entre ellos proporcionando la funcionalidad global.

El flujo de operaciones en la aplicación DEFIDNET sigue los esquemas expuestos en las figuras 5.3 y 5.4.

Figura 5.3: Diagrama de flujo de actividad.

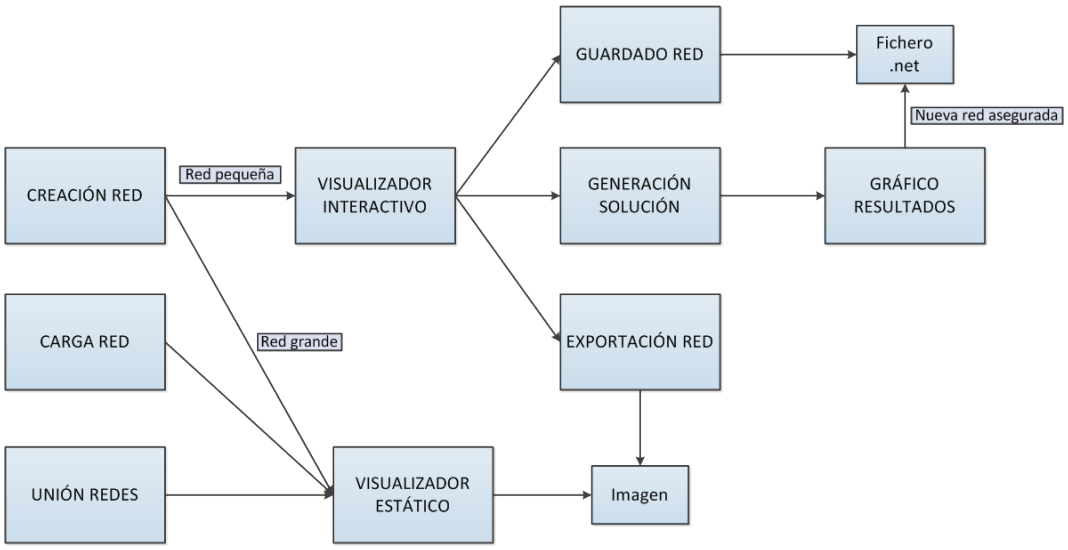

Figura 5.4: Diagrama de flujo de actividad detallado.

5.2. Interfaz gráfica

El diseño de la interfaz gráfica presentada por DEFIDNET pretende ser simple y claro, presentando elementos visuales de líneas sencillas.

Se decidió la utilización de este tipo de interfaz pensando en el objetivo último de la aplicación y sus potenciales usuarios.

DEFIDNET es una aplicación de uso específico, orientada a usuarios con conocimientos y familiarizados con el tema tratado, no a un usuario general. Por tanto, era necesaria una interfaz simple y directa en lugar de una interfaz que resultase atractiva. DEFIDNET se considera una herramienta de uso académico y/o profesional en los cuales prima la consecución correcta de objetivos y eficiencia, no la interfaz como tal. En este caso, la presentación de la aplicación sirve como herramienta para la interacción del usuario con la sistema y para guiar el flujo normal de operaciones, no como un posible reclamo.

A continuación se pasan a explicar las distintas características de la interfaz definida en la fase de diseño del proyecto.

El usuario en primera instancia solo tiene tres posibles acciones a realizar: crear una nueva IDN, cargar una ya existente o unir distintas IDNs en una única. Después de elegir una de las opciones anteriores, es capaz de visualizar u operar con la IDN (no en todos los casos). Por tanto, se decidió la creación de una interfaz inicial con un menú que contuviese dichas opciones especificadas. Además, en esta interfaz se añadiría información referente a la aplicación en su conjunto, como una pestaña de ayuda o de información sobre el proyecto DEFIDNET.

En la siguiente figura 5.5 se observa la interfaz diseñada para el menú principal.

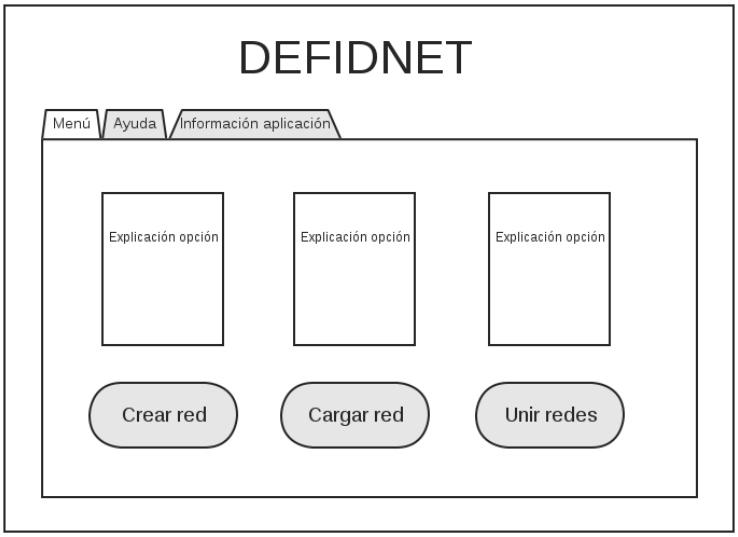

Figura 5.5: *Mockup* del menú principal.

En caso de seleccionar la opción de creación de una nueva IDN aparecería un formulario con las distintas características a definir por el usuario para generarla correctamente.

La figura 5.6 muestra un *mock up* de la interfaz del proceso de creación de una IDN.

Figura 5.6: *Mockup* del formulario de creación de una IDN.

Una vez realizada una de las diferentes opciones iniciales, el repertorio de acciones disponibles aumenta notablemente al poder operar con la IDN actual. Se decidió que uno de los principales elementos, y el eje central de la aplicación, sería el visualizador interactivo de IDNs.

El visualizador interactivo estaría formado por un panel de dibujo y un menú superior destinado a la interacción usuario-aplicación.

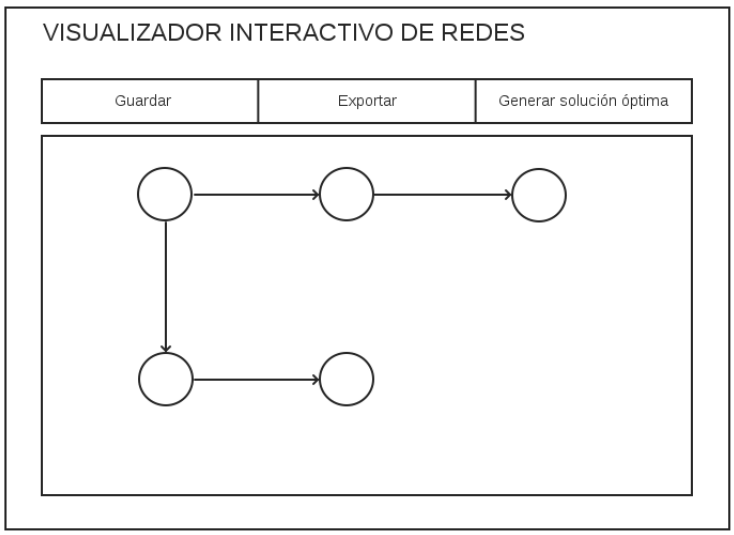

Figura 5.7: *Mockup* del visualizador interactivo de IDNs.

A dicho visualizador se le podrían añadir o modificar nodos mediante un formulario cómo el presentado a continuación.

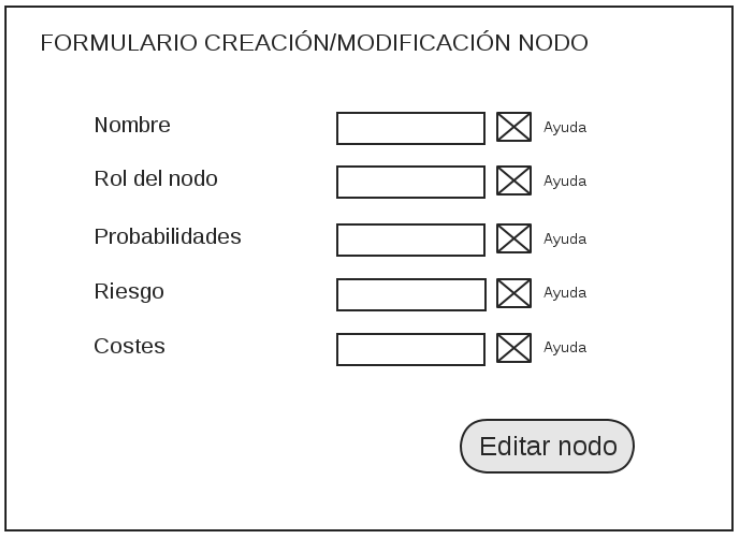

Figura 5.8: *Mockup* del formulario de creación/modificación de un nodo.

Al seleccionar la opción de generar una solución se mostraría al usuario un gráfico con una relación de puntos coste-beneficio obtenidos por el sistema. La figura 5.9

muestra la ventana descrita.

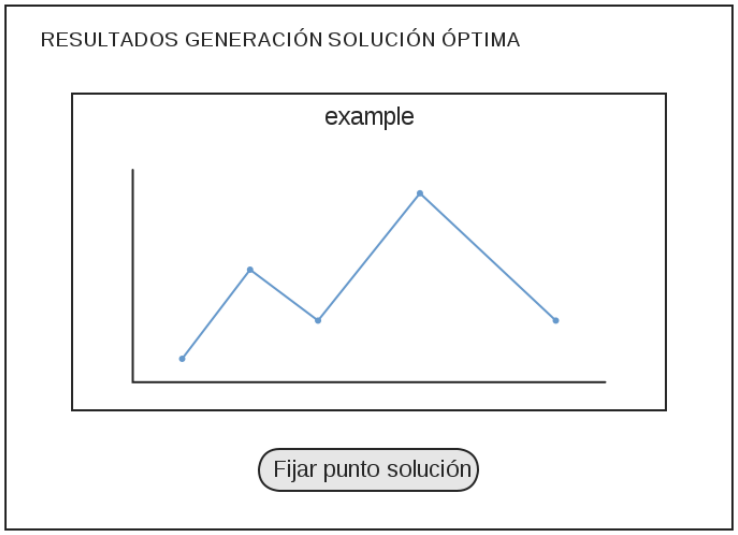

Figura 5.9: *Mockup* de la presentación de resultados de la solución óptima.

Aunque la interfaz final presenta algunas pequeñas modificaciones respecto a la originalmente diseñada debidas a cambios obvios en las necesidades de la aplicación durante el desarrollo del proyecto, en líneas generales se respetó el diseño realizado, consiguiendo una interfaz similar a la prevista.

Capítulo 6

Implementación

6.1. Implementación del sistema

En este capítulo se detalla el proceso de codificación e implementación de la aplicación DEFIDNET para proporcionar una visión más detallada del trabajo llevado a cabo.

A continuación, se exponen los distintos puntos a tratar en los que está dividida esta sección:

- **Entorno de desarrollo**. Se describen las características del entorno de desarrollo utilizado en el proyecto, así como los pasos previos necesarios antes de comenzar la codificación.

- **Diagrama de clases**. Apartado en el que se muestra el diagrama de las principales clases que forman DEFIDNET, permitiendo una visión global y a la vez detallada de la aplicación en su conjunto.

- **Desarrollo de la aplicación**. Parte más extensa y completa referente a la aplicación DEFIDNET de la memoria. Se explica el proceso de codificación y desarrollo, explicando los pasos realizados y cuestiones técnicas. Además, en este punto se describe el proceso de integración de las herramientas externas

utilizadas para la consecución de cada módulo descritas en la sección *2.7. Herramientas utilizadas.*

6.1.1. Entorno de desarrollo

Para la codificación de la aplicación DEFIDNET se ha utilizado Netbeans. Netbeans es un conocido IDE [1] de código abierto que proporciona un entorno de desarrollo para Java, C++ y otros lenguajes de programación.

En un principio se tomaron en cuenta dos posibles alternativas: Netbeans y Eclipse, debido a la familiaridad de la alumna con ambos entornos. Finalmente, se optó por utilizar Netbeans en lugar de Eclipse porque proporciona de manera nativa una herramienta visual para el diseño de interfaces gráficas con Java Swing (sin necesidad de instalación de *plugins*) en la cual se pueden diseñar mediante *drag and drop*[2] visualizando en todo momento el resultado.

Eclipse, además mostró problemas de compatibilidad con el plugin encargado de proporcionar esa funcionalidad, por lo cual fue descartado.

Como paso previo a la codificación, se importaron los paquetes del DEFIDNET original en Netbeans, siendo la mayoría de ellos pertenecientes a ECJ. En el punto *6.1.3.1 Organización de DEFIDNET* se puede consultar el esquema de los distintos paquetes de los que se compone finalmente la aplicación.

[1] *Integrated Development Environment.*

[2] *Drag and drop.* Arrastrar y soltar.

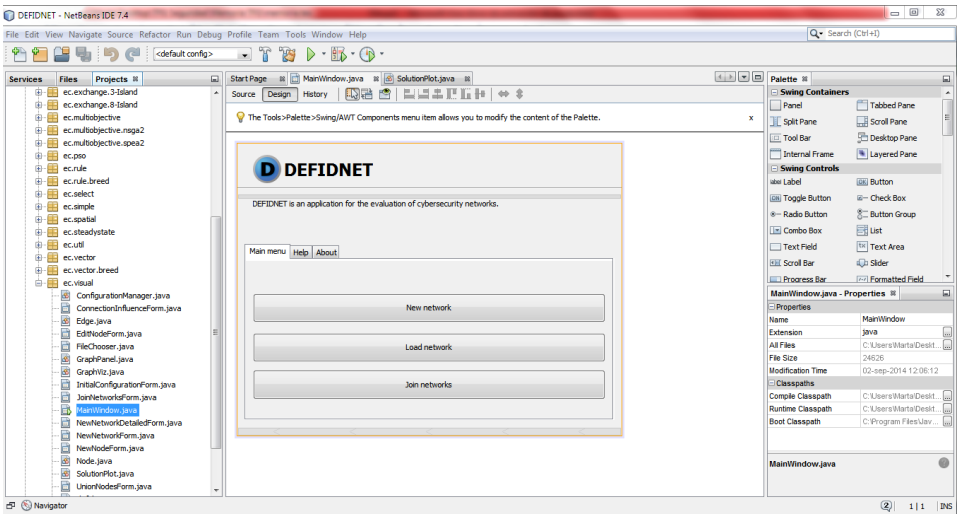

Figura 6.1: Captura del IDE Netbeans.

6.1.2. Diagrama de clases

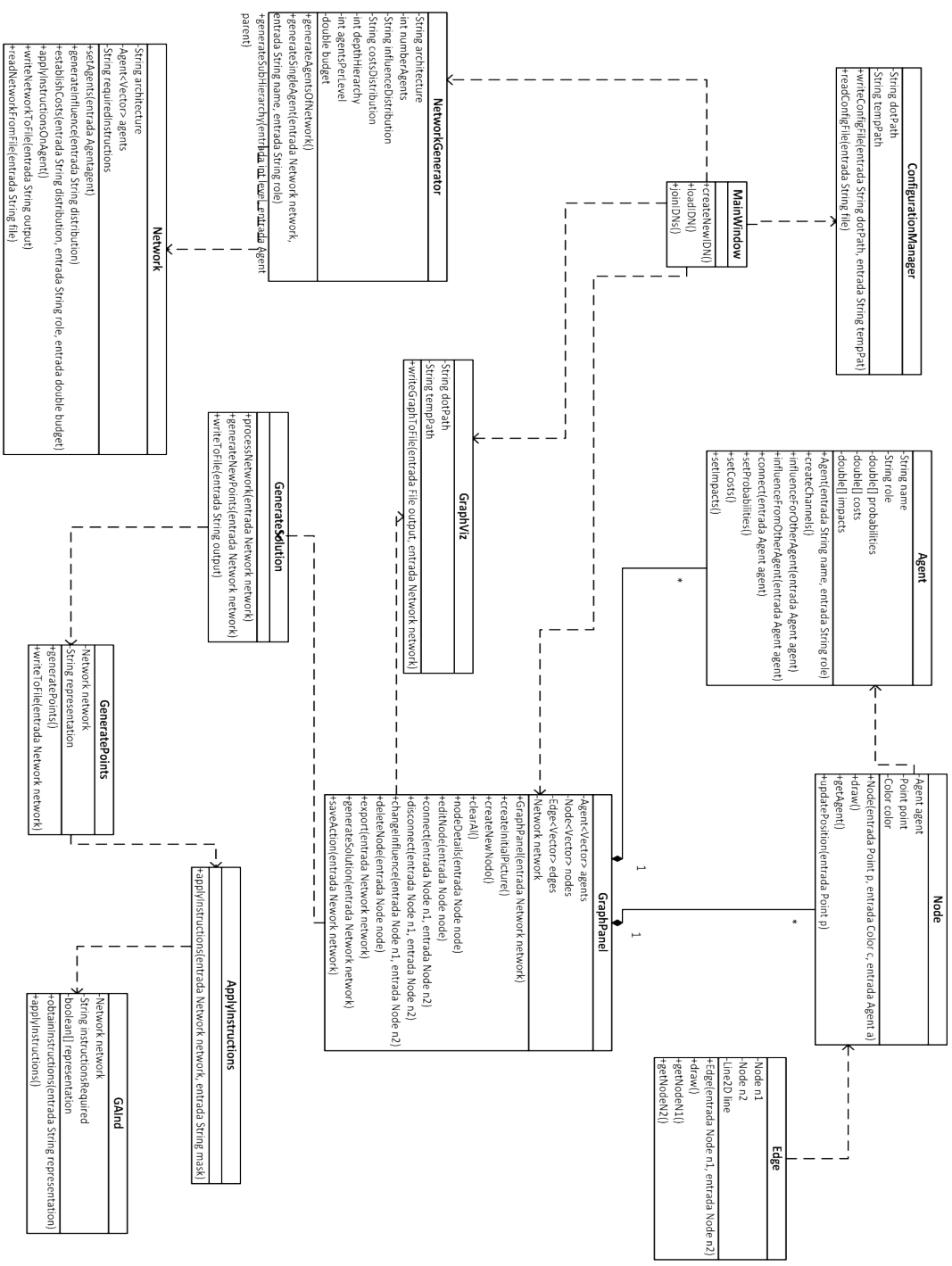

Figura 6.2: Esquema general de clases.

En el diagrama de clases de la figura anterior (6.2) se representan las clases que proveen la funcionalidad de DEFIDNET. Han quedado excluidas las pertenecientes a la vista de la aplicación; sin embargo, *GraphPanel* y *MainWindow* sí se han tenido en cuenta (puesto que tienen una importancia global mucho mayor que el resto).

Para visualizar correctamente la imagen se recomienda seleccionar la opción de orientación apaisada en el visor de PDFs y hacer uso del *zoom*.

6.1.3. Desarrollo de la aplicación

En este apartado se describe el proceso de codificación de DEFIDNET mediante una visión más técnica que las utilizadas hasta el momento, modularizada para hacer más sencilla su estructura y su comprensión por parte del lector. Se explican también algunos problemas sufridos durante la implementación y cómo se resolvieron.

En la figura 5.2 se puede ver el gráfico del esquema inicial de la aplicación realizado en la fase de diseño (capítulo *5. Diseño*). A dicho gráfico se le han añadido las herramientas externas necesarias para la consecución de cada módulo, de tal manera que se puede observar fácilmente la estructura general de DEFIDNET.

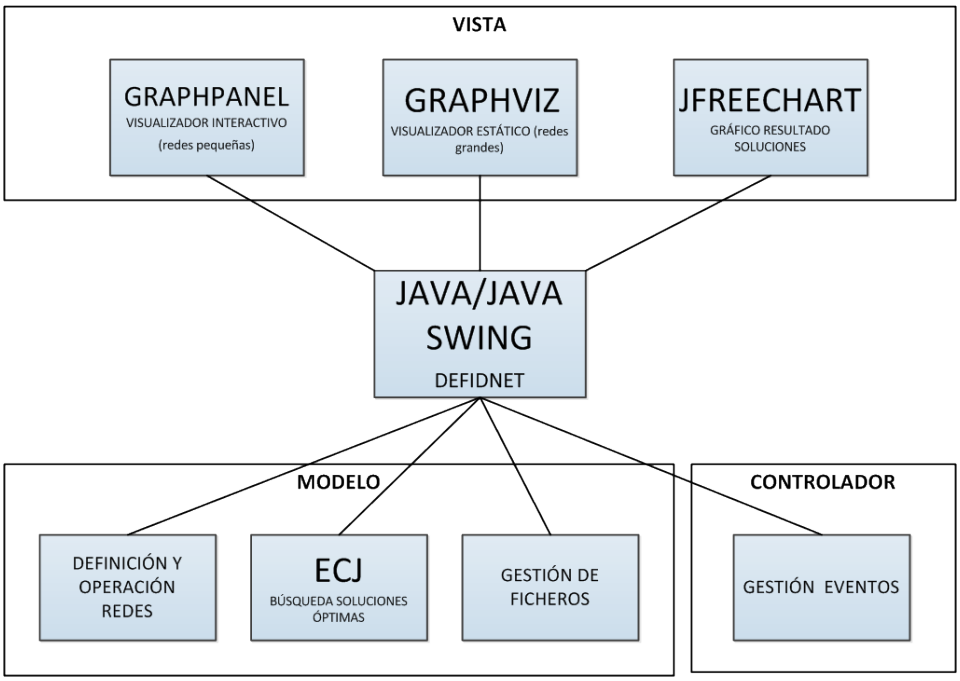

Figura 6.3: Diagrama de módulos de la aplicación DEFIDNET.

Varios módulos requieren de herramientas externas que simplifiquen el trabajo y aporten características que de otro modo sería difícil conseguir; sin embargo, otros no requieren de ningún elemento externo, basándose en las funcionalidades proporcionadas por el propio lenguaje Java y la librería java Swing.

En los siguientes puntos, se desglosan las tareas llevadas a cabo durante la implementación, divididas según los módulos expuestos.

6.1.3.1. Organización de DEFIDNET

DEFIDNET sigue un modelo MVC (*Modelo-Vista-Controlador*) mediante el cual se separa la capa de presentación del modelo.

La organización en distintos paquetes permite una estructuración lógica de la aplicación. En la figura 6.4 se pueden consultar los paquetes de los que se compone.

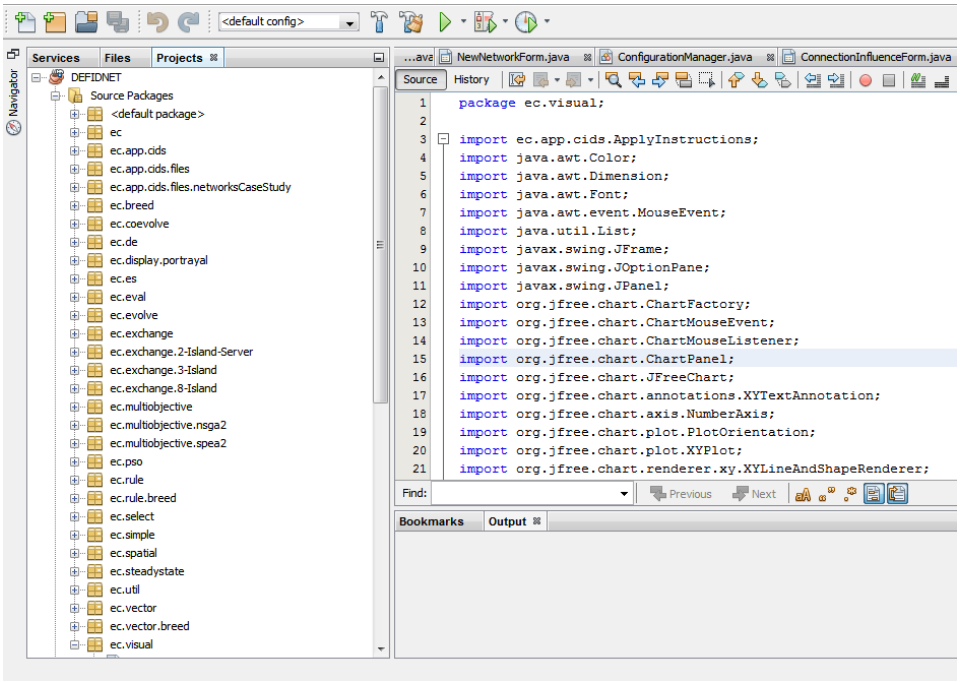

Figura 6.4: Estructura de los paquetes de DEFIDNET.

Se puede observar que todos los paquetes comienzan por ec (de *Evolutionary Computation*). Esto se debe a que la mayoría pertenecen a la herramienta ECJ (véase *2.7 Herramientas utilizadas*). Los dos paquetes realmente relevantes y que implementan DEFIDNET son ec.app.cids y ep.visual. A continuación, se presenta la tabla 6.1 con sus clases y una breve descripción.

Paquete	Descripción breve	Clases
ec.app.cids	Clases que implementan el modelo.	*Agent*
		ApplyInstructions
		CIDS
		ConfigurationManager
		GAInd
		GeneratePoints
		GenerateSolution
		Network
		NetworkGenerator
ec.visual	Clases que implementan la vista de la aplicación.	*ConnectionInfluenceForm*
		Edge
		EditNodeForm
		GraphPanel
		GraphViz
		InitialConfigurationForm
		JoinNetworksForm
		MainWindow
		NewNetworkDetailedForm
		NewNetworkForm
		NewNodeForm
		Node
		NodeDetails
		SolutionPlot
		UnionNodesForm

Tabla 6.1: Descripción de los paquetes relevantes de DEFIDNET.

El paquete ec.app.cids contiene el modelo de DEFIDNET, en el cual se implementa la funcionalidad del programa; como es la exportación a imágenes, la escritura de

una IDN en un fichero de sistema o la creación de un nodo en la red, por ejemplo. Estos métodos son llamados desde la vista en respuesta a la interacción del usuario mediante eventos.

El paquete ec.visual contiene las clases de presentación; es decir, de la interfaz gráfica. Incluye las distintas ventanas, formularios y paneles con los que el usuario debe interactuar. Los eventos implementados en estas clases (los cuales forman el controlador del patrón MVC) se encargan de realizar llamadas a los métodos que representan el modelo.

El resto de paquetes contienen clases necesarias para el correcto funcionamiento de ECJ, sistema de computación evolutiva que genera las soluciones óptimas para la IDN.

6.1.3.2. Interfaz general

La interfaz general de DEFIDNET es un punto muy importante en el desarrollo de la aplicación, puesto que sirve de elemento comunicador entre las funcionalidades existentes y entre el usuario y el programa. La secuencia de ventanas permite al usuario navegar entre las distintas operaciones a través de diálogos y formularios.

Para su realización se ha utilizado Java Swing, una librería gráfica de Java explicada más detalladamente en la sección *2.7. Herramientas utilizadas*. En algunos casos también ha sido necesario el uso de Java AWT, parte de la API estándar de Java para desarrollar interfaces.

El desarrollo de la interfaz siguiendo el modelo diseñado previamente ha resultado sumamente sencillo gracias al editor visual de Netbeans, que permite disponer los diferentes elementos sin tener que codificar manualmente su tamaño, coordenadas y demás características. Si bien, por supuesto, en la mayoría de los casos se

ha requerido de opciones más específicas y ha sido necesario editar el código auto-generado por Netbeans.

La comunicación entre las distintas ventanas se ha realizado mediante la instanciación de clases, recibiendo cada una de ellas en su constructor o en otros métodos la información requerida desde la clase anterior.

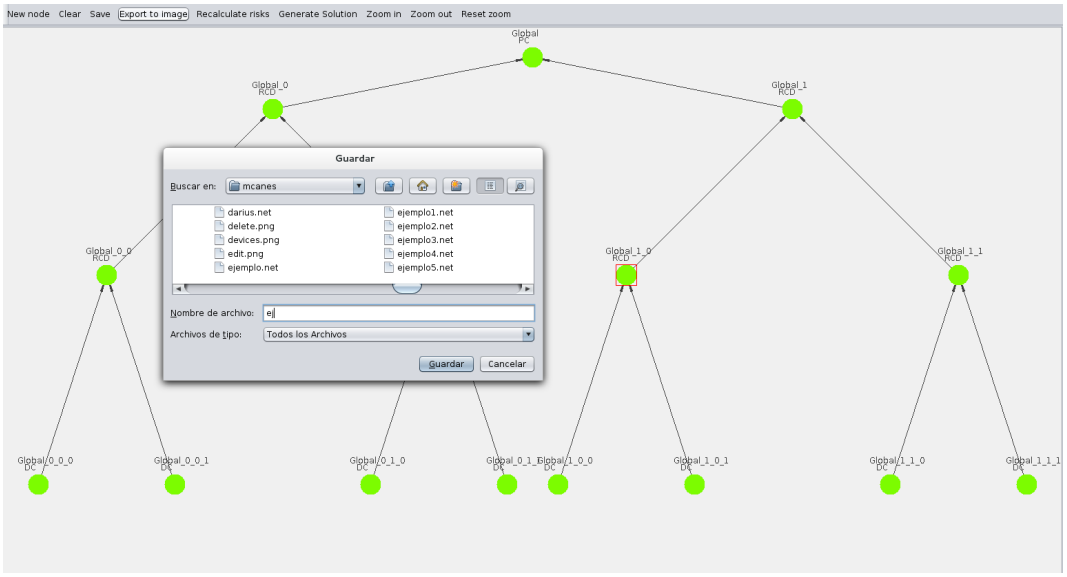

Figura 6.5: Ejemplo de ventana de DEFIDNET.

En el *Anexo I. Manual de usuario* se pueden ver distintas capturas de la interfaz gráfica de la aplicación.

6.1.3.3. Gestión de eventos

Este módulo se ha llevado a cabo mediante Java Swing y su modelo de eventos. Éstos, representan el controlador de la aplicación y llaman al modelo, que ejecuta la funcionalidad asociada. Después, actualizan la interfaz con la información recibida. Para saber más sobre el patrón utilizado véase la sección *5.1. Arquitectura*.

Un evento es disparado por una acción del usuario sobre la interfaz de DEFIDNET, como puede ser pulsar un botón o seleccionar una opción de un menú desplegable.

Por ejemplo, el botón *New IDN* tiene asociado un evento. Cuando el usuario lo pulsa, dicho evento es disparado y se ejecuta la funcionalidad que contiene mediante la clase *ActionEvent* de Java AWT. Como se puede ver en la figura 6.6, en este caso el código es muy sencillo: se abre una nueva ventana (el formulario de creación de una IDN).

```
private void new_network_btnActionPerformed(java.awt.event.ActionEvent evt) {
        NewNetworkForm newNetworkForm = new NewNetworkForm();
        newNetworkForm.setVisible(true);
}
```

Figura 6.6: Ejemplo de código de un evento.

6.1.3.4. Gestión de ficheros

El almacenamiento permanente de IDNs así como de configuraciones y parámetros para el correcto funcionamiento de DEFIDNET ha sido gestionado mediante ficheros de sistema.

Cuando el usuario interactúa con una IDN y desea almacenarla; ésto se hace guardándola en un fichero de extensión .net utilizando el método *writeNetworkToFile()* de la clase *Network*. Así mismo, cuando desea cargarla de nuevo en la aplicación ese fichero es leído y recuperado (mediante *readNetworkFromFile()*, de la clase *Network* también).

Del mismo modo ocurre cuando se desea exportar una IDN a imagen. Dicha IDN es almacenada en un fichero .dot con las características necesarias para poder ser leído por GraphViz, procesado y finalmente exportado a un fichero de extensión .gif.

La generación de soluciones también opera con diversos ficheros de sistema, los cuales se explican más adelante en el punto *6.1.3.8 Generación de soluciones*.

Por tanto, se puede concluir que la gestión de ficheros de sistema es un punto básico de DEFIDNET, sin el cual no sería posible el almacenamiento no volátil de información, puesto que no se implementa una base de datos. El almacenamiento es llevado a cabo enteramente mediante ficheros.

El modelo de la aplicación es el encargado de interactuar con la información permanente, escribiendo en los ficheros y más tarde leyéndolos mediante las clases nativas del API de Java: *PrintWriter* y *BufferedReader* (las cuales son invocadas por los generadores de eventos en los casos en los que es necesario).

6.1.3.5. Definición y operaciones sobre IDNs

Las IDNs pueden ser creadas, guardadas, cargadas, combinadas, modificadas, exportadas y aseguradas (utilizando la aplicación de contramedidas) mediante DEFIDNET.

Para crear una IDN es necesario especificar una serie de datos relativos a ésta:

- Arquitectura de la IDN.
- Distribución de las influencias entre nodos.
- Número de nodos.
- Probabilidad de conexión (sólo en las de arquitectura distribuida parcialmente).
- Número de niveles (sólo en las jerárquicas).
- Número de nodos por nivel (sólo en las jerárquicas).

Tras la especificación general de la IDN, es necesario introducir la información relativa a las características de cada uno de los roles que la conforman:

- Probabilidad de ser atacado.

- Porcentaje de nodos atacados.
- El coste total de aplicar contramedidas en la IDN.
- Distribución de los costes entre los distintos nodos.
- Impacto de cada uno de los tipos de ataque (explicados en el punto *2.3.4. Tipos de ataque*).

Las IDNs son guardadas en formato .net mediante los ficheros de sistema especificados en el punto anterior. Para ser cargadas de nuevo en la aplicación simplemente se debe seleccionar el fichero en el formulario de carga de la IDN.

La unión de IDNs consiste en la combinación de varias IDNs en una única. Cada IDN combinada tiene uno o varios nodos que actúan como unión. La red resultante es almacenada en un fichero .net como una IDN normal, con la particularidad de que tiene una arquitectura del tipo *joined*.

Así mismo, mediante el visualizador interactivo explicado en el siguiente punto (punto *6.1.3.6 Visualizador interactivo*) es posible añadir nodos, editar o borrar los ya existentes y modificar las conexiones.

La exportación a imagen de una red es llevada a cabo por la herramienta GraphViz. Se puede encontrar más información acerca del proceso en el punto *6.1.3.7. Visualizador estático.*

Y por último, como ocurre con la exportación, el proceso de generación de soluciones y posterior asegurado de la IDN es descrito en los puntos *6.1.3.8. Generación de soluciones* y *6.1.3.9. Presentación de resultados.*

6.1.3.6. Visualizador interactivo

GraphPanel es la herramienta utilizada para desarrollar el visualizador interactivo de DEFIDNET. Dicho visualizador representa la funcionalidad añadida más

importante del programa, que es permitir al usuario visualizar y modificar IDNs (50 nodos o menos) de pequeño tamaño mediante figuras interactivas.

GraphPanel se compone visualmente de dos elementos, el menú superior y el panel de dibujo. Sin embargo, cuenta con otro menú flotante asociado a cada figura del panel de dibujo, individual para cada nodo.

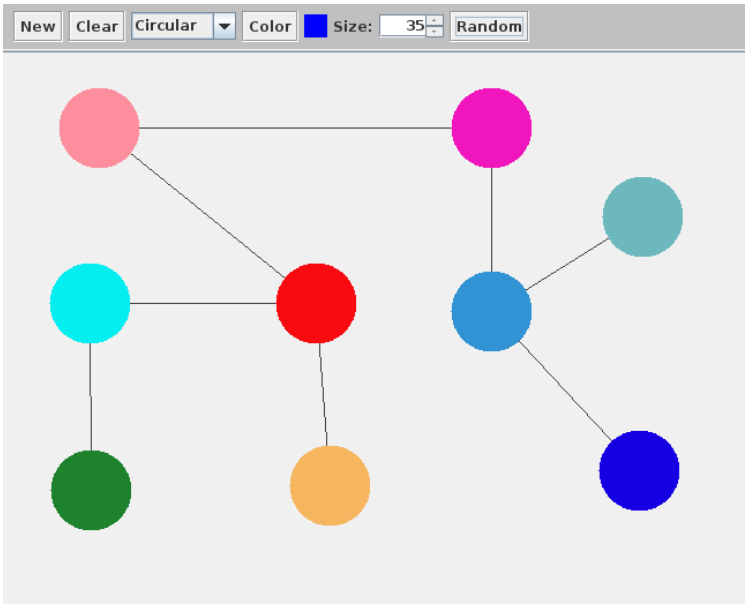

Figura 6.7: Captura de GraphPanel original.

La clase de GraphPanel se compone de varias subclases que heredan de *AbstractAction* y que implementan la funcionalidad de cada una de las operaciones disponibles en los menús. Cuando un usuario pulsa un botón con un evento asociado, éste llama a la subclase correspondiente (la cual ejecuta las instrucciones de la operación especificada). De esta manera, se consigue un mayor grado de modularidad y resulta mucho más sencilla la modificación de las distintas partes del código.

En la siguiente figura (figura 6.8) se puede ver un ejemplo de las subclases descritas en el párrafo anterior relativo a la visualización de los detalles de un nodo.

```
public class NodeDetailsAction extends AbstractAction {
    public NodeDetailsAction(String name) {
        super(name);
    }
    public void actionPerformed(ActionEvent e) {
        for (Node n : nodes) {
            if (n.isSelected()) {
                JFrame frame = new JFrame();
                NodeDetails nodeDetails = new NodeDetails(n.getAgent(), frame,true);
                nodeDetails.setVisible(true);
            }
        }
        repaint();
    }
}
```

Figura 6.8: Subclase de GraphPanel.

Los cambios necesarios sobre GraphPanel, dentro su complejidad, han sido intuitivos y coherentes con el código de partida, gracias a que el código inicial del programa creado por el Dr. Matthews tiene una organización estructurada y fácilmente modificable.

En primer lugar, fue necesario realizar una correspondencia entre las clases de la vista y el modelo. Es decir, entre las figuras encargadas de representar a los nodos (círculos) y las conexiones (líneas dirigidas) de la IDN, y los nodos y conexiones propiamente dichos.

En la práctica, y por motivos de continuidad con el código anterior, la clase *Agent* es la encargada de definir un nodo con sus atributos, mientras que la clase *Node* implementa la figura del círculo que lo representa. La nomenclatura utilizada puede dar pie a errores de comprensión, puesto que los nodos pasan a llamarse agentes y sus figuras son las que realmente utilizan su nombre. Ésto se debe a que se decidió respetar las definiciones aplicadas en el DEFIDNET original, si bien no eran las más adecuadas para este proyecto.

Por su parte, la clase *Edge* representa visualmente las conexiones entre nodos, las cuales no tienen una clase propia en el modelo, sino que se definen mediante

atributos de la clase *Agent*.

Al menú superior se han añadido nuevas opciones y el modo de dibujado de los nodos se ha modificado para responder a los esquemas de las distintas arquitecturas existentes. Tras la creación de una IDN mediante los formularios necesarios, el visualizador interactivo de DEFIDNET muestra un esquema inicial de la IDN, que el usuario puede modificar a su gusto posteriormente.

El usuario puede mover los nodos en el espacio, editar sus características, eliminarlos de la IDN o crear nuevos. Así mismo, se pueden modificar las conexiones entre ellos. La mayoría de estas opciones no formaban parte del GraphPanel original.

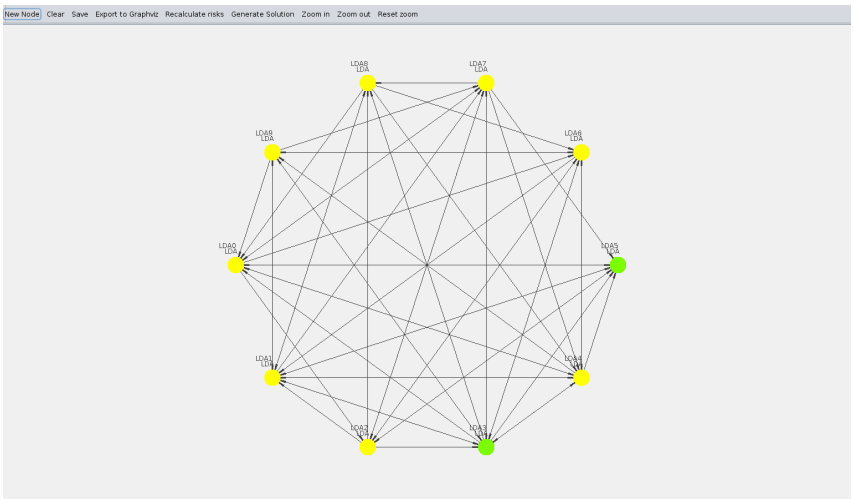

Figura 6.9: Captura de GraphPanel adaptado.

Para mejorar la visualización, se han implementado las funciones de *Zoom in*(acercar el grafo) y *Zoom out*(alejar el grafo). En este proceso se sufrieron problemas referentes a la actualización de las distintas figuras para modificar su tamaño y realizar correctamente el *zoom*. Finalmente, esta funcionalidad se consiguió implementar correctamente.

6.1.3.7. Visualizador estático

El visualizador estático se utiliza para poder ver IDNs de tamaño superior a 50 nodos. Sin embargo, no permite modificarlas. Para realizar esta acción deberá utilizarse la edición manual de ficheros de sistema (se puede consultar más información acerca de este proceso en el punto *12. DEFIDNET en modo manual* del manual de usuario).

El visualizador estático no es un elemento único. Es el resultado de la combinación de un proceso de exportación a imagen de la IDN llevado a cabo por GraphViz (operación que puede realizarse con cualquier IDN, sea del tamaño que sea) y de la visualización de dicha imagen a través del visor de imágenes por defecto en el sistema operativo.

Graphviz ofrece numerosas características y es una herramienta muy potente; sin embargo, no ofrece integración directa con aplicaciones desarrolladas en Java de tal manera que el proceso de conversión a imagen sea un proceso interno de la aplicación. Para poder llevar a cabo esto, se utilizó una API intermedia desarrollada por Laszlo Szathmary [21].

Con esta API se requiere de la definición de un fichero de configuración para poder hacer uso de GraphViz de manera interna en el programa sin necesidad de ejecutar comandos manualmente. Dicho fichero recibe el nombre de app.config y contiene información sobre las rutas de dot.exe (en el caso de Windows) y de dot.sh (en Linux) de la carpeta de GraphViz y sobre el directorio temporal deseado por el usuario en el cual se almacenará la información del proceso de conversión.

Al iniciarse, DEFIDNET comprobará la existencia del fichero. En caso de no encontrarlo o no poder leerlo, mostrará un formulario al usuario para la especificación de la información necesaria. La aplicación consta de una opción (*GraphViz Settings*) para modificarlo posteriormente mediante la interfaz gráfica.

Así mismo, el fichero app.config puede crearse y editarse manualmente, constando únicamente de dos líneas.

```
CONFIGURATION FILE - DEFIDNET APPLICATION
Graphviz path=/usr/bin/dot
Temporal directory path=/tmp
```

Figura 6.10: Fichero de configuración inicial de GraphViz.

Este fichero es importante para otras funcionalidades de DEFIDNET, por lo que es necesaria su definición para disponer de todas las opciones y operaciones posibles.

Puesto que GraphViz es independiente de la aplicación desarrollada en el presente proyecto, es necesario que el usuario descargue esta herramienta de manera separada y especifique la información descrita anteriormente una vez este instalada en el sistema.

En la figura 6.11 se puede visualizar el resultado de una IDN exportada a imagen mediante GraphViz.

Figura 6.11: Captura de IDN exportada a imagen mediante GraphViz.

6.1.3.8. Generación de soluciones

La generación de soluciones se basa en la búsqueda multiobjetivo, que permite encontrar una serie de soluciones que representan el valor óptimo coste-beneficio (siendo el beneficio en este caso el riesgo mitigado por las contramedidas) para asegurar una IDN dada. ECJ es la herramienta encargada del proceso de generación de soluciones proporcionando la implementación de la búsqueda evolutiva mediante el algoritmo SPEA2.

La búsqueda multiobjetivo consiste en optimizar los elementos de una función objetivo, devolviendo una serie de puntos conocidos como Frente de Pareto. Cada uno de esos puntos representa el valor óptimo de tal manera que no es posible

conseguir una mejora en uno de los elementos de la función sin perjudicar al otro [3].

El algoritmo SPEA2 es una version mejorada del algoritmo SPEA (*The Strength Pareto Evolutionary Algorithm*) creado en 1999 [13]. Permite encontrar el Frente de Pareto, obteniendo los valores óptimos para una función dada [12].

La información de configuración de ECJ se encuentra en los ficheros cids.params y spea2.params (localizados en el paquete ec.app.cids), y si bien están especificados unos parámetros por defecto con los cuales el usuario puede perfectamente ejecutar la aplicación y obtener buenos resultados, éstos son configurables manualmente para editar sus características, como por ejemplo el número de generaciones del algoritmo SPEA2 (véase la sección *12. DEFIDNET en modo manual* del manual de usuario).

```
# Copyright 2010 by Sean Luke and George Mason University
# Licensed under the Academic Free License version 3.0
# See the file "LICENSE" for more information

parent.0 = ../../multiobjective/spea2/spea2.params


# SPEA2 varies in its archive size, and in ECJ the archive size is part of the
# population, as opposed to NSGA-II, which temporarily builds its own archive
# that is EXTERNAL to the population.  This means that in order to generate
# an equivalent number of individuals to NSGA-II, we'll need to have a larger
# population size in SPEA2.  Specifically, the population size should be
# the NSGA-II size plus the desired archive size (number of elites).
#
# For more information, see discussion in the README files in the SPEA2 and
# NSGA-II packages
#
# constants from  "Zitzler, E., Deb, K., and Thiele, L., 2000,
# Comparison of Multiobjective Evolutionary Algorithms: Empirical Results,
# Evolutionary Computation, Vol. 8, No. 2, pp173-195."
#

pop.subpop.0.size =                     250
breed.elite.0 =                         5
generations =                           10

# SPEA2's pipeline:
pop.subpop.0.species.pipe                       = ec.vector.breed.VectorMutationPipeline
pop.subpop.0.species.pipe.likelihood            = 1.0
pop.subpop.0.species.pipe.source.0              = ec.vector.breed.VectorCrossoverPipeline
pop.subpop.0.species.pipe.source.0.likelihood   = 0.9
pop.subpop.0.species.pipe.source.0.source.0     = ec.multiobjective.spea2.SPEA2TournamentSelection
pop.subpop.0.species.pipe.source.0.source.1     = same
select.tournament.size                          = 15
```

Figura 6.12: Extracto del fichero spea2.params.

6.1.3.9. Presentación de resultados

Una vez finalizada la búsqueda multiobjetivo, los resultados son mostrados mediante un gráfico Frente de Pareto generado con JFreeChart (véase la figura 6.13).

Los distintos puntos representan los paretos. Un pareto es una solución al problema multiobjetivo, de tal manera que no existe otra solucion que mejore uno de los objetivos sin empeorar el otro. Es decir, cada punto representa la mejor combinación de coste y riesgo mitigado (beneficio).

La interacción entre el usuario y el gráfico ha sido posible mediante la utilización de los métodos *getRangeCrosshairValue()* y *getDomainCrosshairValue()* proporcionados por JFreeChart, que permiten obtener el punto más cercano a las coordenadas seleccionadas por el usuario.

Una vez seleccionado el punto deseado por el usuario se aplican contramedidas para asegurar la IDN según el coste-beneficio especificados en ese punto.

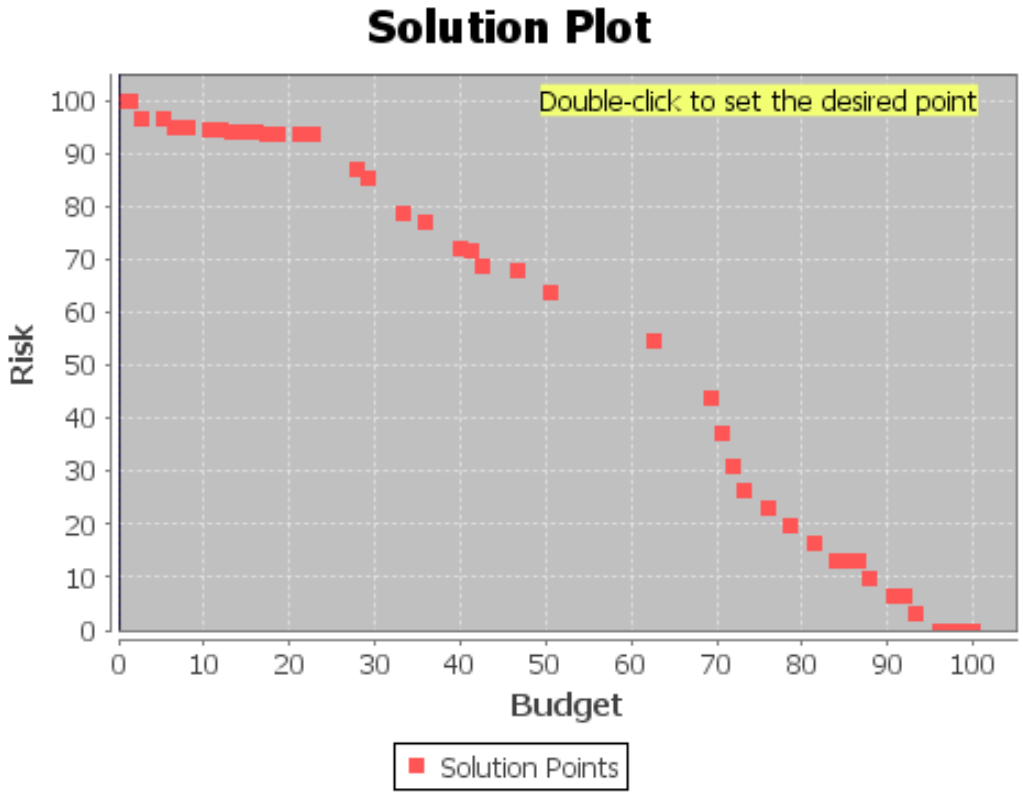

Figura 6.13: Captura de JFreeChart.

Capítulo 7

Pruebas y evaluación del sistema

7.1. Pruebas

La especificación y realización de pruebas es una fase sumamente importante en el desarrollo de un proyecto. Las pruebas permiten verificar que el software cumple los requisitos obtenidos al inicio del desarrollo y está listo para ser utilizado.

En las tablas situadas a continuación se pasa a describir la relación de pruebas llevadas a cabo para comprobar que la aplicación DEFIDNET satisface las necesidades para las que fue creada, así como verificar que no presenta errores de codificación o fallos.

Las pruebas han de comprobar que la aplicación responde de manera adecuada en el caso de que el usuario introduzca valores válidos o realice operaciones de forma correcta; pero también se ha de verificar que el comportamiento es el esperado cuando el usuario no actúa de la manera adecuada introduciendo información no válida.

Las pruebas son descritas mediante tablas con el formato descrito a continuación:

- **ID Prueba**. Identificador de la prueba. Sigue el formato PS-XXX, donde XXX indica el número de prueba.
- **Título**. Título breve y significativo que permite identificar fácilmente la funcionalidad que la prueba verifica.
- **Descripción**. Descripción más extensa que el título que explica de manera más detallada la prueba a llevar a cabo y sus características.
- **Secuencia de pasos**. Listado de acciones necesarias para llevar a cabo correctamente la prueba definida.
- **Resultado esperado**. Salida considerada como determinante para considerar la prueba superada con éxito.

A continuación, se expone la relación de pruebas realizadas sobre la aplicación:

ID Prueba	PS-001	
Título	Comprobación inicial del fichero de configuración de GraphViz.	
Descripción	La primera vez que se inicia la aplicación, ésta debe comprobar si existe el fichero de configuración. En caso negativo, debe mostrar un formulario al usuario para su creación.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
Resultado esperado	La aplicación muestra el formulario de generación del fichero de configuración de GraphViz.	

Tabla 7.1: Prueba PS-001.

ID Prueba	PS-002	
Título	Creación del fichero de configuración.	
Descripción	Al completar el formulario para la creación del fichero de configuración debe crearse el fichero app.config en el directorio raíz de la aplicación.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	1	El usuario completa correctamente el formulario de creación del fichero de configuración.
Resultado esperado	El fichero app.config es creado en el directorio raíz de la aplicación.	

Tabla 7.2: Prueba PS-002.

ID Prueba	PS-003	
Título	Modificación del fichero de configuración.	
Descripción	Al completar el formulario para la modificación del fichero de configuración debe editarse con la nueva información el fichero *app.config* en el directorio raíz de la aplicación.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario rellena correctamente el formulario para la modificación del fichero de configuración.
Resultado esperado	El fichero app.config en el directorio raíz de la aplicación es modificado.	

Tabla 7.3: Prueba PS-003.

ID Prueba	PS-004	
Título	Creación de una nueva IDN.	
Descripción	Al completar los formularios de creación de una nueva IDN, ésta debe crearse satisfactoriamente y ser visualizada por el usuario.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario introduce datos válidos en los formularios de creación de una IDN.
Resultado esperado	La IDN es creada y visualizada por el usuario.	

Tabla 7.4: Prueba PS-004.

ID Prueba	PS-005	
Título	Carga de una IDN.	
Descripción	Al seleccionar el fichero válido que almacena una IDN ya existente, ésta debe ser visualizada por el usuario.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario selecciona un fichero válido que contenga una IDN.
Resultado esperado	La IDN es cargada en la aplicación y visualizada por el usuario.	

Tabla 7.5: Prueba PS-005.

ID Prueba	PS-006	
Título	Unión de IDNs.	
Descripción	Al completar los formularios de unión de IDNs, éstas deben combinarse satisfactoriamente y ser visualizadas por el usuario.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario introduce valores válidos en los formularios de combinación de IDNs, seleccionando al menos un nodo de unión para cada red.
Resultado esperado	Las IDNs son combinadas correctamente y se muestran al usuario. La influencia entre los nodos de unión debe estar equitativamente repartida de tal forma que todas las conexiones sumen 1.	

Tabla 7.6: Prueba PS-006.

ID Prueba	PS-007	
Título	Visualización de una IDN de pequeño tamaño.	
Descripción	Las IDNs de pequeño tamaño (menos o igual a 50 nodos) deben visualizarse en el visualizador interactivo de IDNs, siempre que no tengan arquitectura *joined* (es decir, que sean el resultado de la unión entre 2 o más IDNs).	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario crea/carga una IDN válida de 50 nodos o menos.
Resultado esperado	La red es visualizada en el visualizador interactivo.	

Tabla 7.7: Prueba PS-007.

ID Prueba	PS-008	
Título	Visualización de una IDN de gran tamaño.	
Descripción	Las redes de tamaño grande (más de 50 nodos) o de tipo *joined* deben ser visualizadas en el visualizador estático.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario crea/carga una IDN válida de más de 50 nodos o de tipo *joined*.
Resultado esperado	La red es visualizada mediante el visualizador estático.	

Tabla 7.8: Prueba PS-008.

ID Prueba	PS-009	
Título	Inserción de un nuevo nodo en la IDN.	
Descripción	Al completarse el formulario de inserción de un nuevo nodo en la IDN, éste debe añadirse correctamente.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario crea/carga una IDN de pequeño tamaño para ser visualizada en el visualizador interactivo.
	3	El usuario introduce valores válidos en el formulario de inserción de un nuevo nodo.
Resultado esperado	El nodo es creado e insertado correctamente en la IDN. Aparece en el visualizador interactivo.	

Tabla 7.9: Prueba PS-009.

ID Prueba	PS-0010	
Título	Modificación de un nodo de la IDN.	
Descripción	Al rellenar correctamente el formulario de modificación de un nuevo nodo de la IDN, éste debe modificarse correctamente.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario crea/carga una IDN de pequeño tamaño para ser visualizada en el visualizador interactivo.
	2	El usuario introduce valores válidos en el formulario de modificación de un nodo.
Resultado esperado	El nodo es modificado y actualizado correctamente en la IDN. Aparece en el visualizador interactivo con las nuevas características.	

Tabla 7.10: Prueba PS-010.

ID Prueba	PS-011	
Título	Eliminación de un nodo de la IDN.	
Descripción	Al seleccionar la opción de eliminar un nodo de la IDN, éste debe ser borrado junto con sus conexiones a otros nodos.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario crea/carga una IDN de pequeño tamaño para ser visualizada en el visualizador interactivo.
	3	El usuario selecciona la opción de eliminación de un nodo.
Resultado esperado	El nodo es eliminado y sus conexiones con otros nodos de la IDN borradas.	

Tabla 7.11: Prueba PS-011.

ID Prueba	PS-012	
Título	Conexión de dos nodos de la IDN.	
Descripción	Al seleccionar dos nodos y pulsar la opción *Connect*, ambos nodos deben quedar conectados, pasando el primer nodo seleccionado al segundo nodo seleccionado la influencia especificada en el formulario.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario crea/carga una IDN de pequeño tamaño para ser visualizada en el visualizador interactivo.
	3	El usuario selecciona dos nodos y pulsa la opción *Connect*.
Resultado esperado	Los nodos quedan conectados, pasando el primer nodo conectado su influencia al segundo nodo conectado (flecha dirigida).	

Tabla 7.12: Prueba PS-012.

ID Prueba	PS-013	
Título	Conexión de nodos de la IDN (selección de un número distinto de dos nodos).	
Descripción	Al seleccionar el usuario un único nodo o más de dos nodos para conectar se debe mostrar un mensaje de error.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario crea/carga una IDN de pequeño tamaño para ser visualizada en el visualizador interactivo.
	3	El usuario selecciona o uno o más de dos nodos y pulsa la opción *Connect* del submenú (botón derecho del ratón sobre uno de los nodos seleccionados).
Resultado esperado	La aplicación muestra un mensaje de error.	

Tabla 7.13: Prueba PS-013.

ID Prueba	PS-014	
Título	Desconexión de dos nodos de la IDN.	
Descripción	Al seleccionar dos nodos de la IDN y pulsar la opción *Disconnect*, la conexión entre ambos en el sentido especificado debe ser borrada.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario crea/carga una IDN de pequeño tamaño para ser visualizada en el visualizador interactivo.
	3	El usuario selecciona dos nodos y pulsa la opción *Disconnect* del submenú (botón derecho del ratón sobre uno de los nodos seleccionados).
Resultado esperado	La aplicación elimina la conexión entre los dos nodos.	

Tabla 7.14: Prueba PS-014.

ID Prueba	PS-015	
Título	Modificación de la influencia de conexión entre dos nodos de la IDN.	
Descripción	Al seleccionar dos nodos de la IDN y pulsar la opción *Change influence* se debe mostrar un formulario en el cual el usuario especificará la nueva influencia entre los dos nodos. Es modificada la influencia que ejerce el primer nodo seleccionado sobre el segundo.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario crea/carga una IDN de pequeño tamaño para ser visualizada en el visualizador interactivo.
	3	El usuario selecciona dos nodos y pulsa la opción *Change influence* del submenú (botón derecho del ratón sobre uno de los nodos seleccionados).
Resultado esperado	La influencia de conexión entre los dos nodos es modificada.	

Tabla 7.15: Prueba PS-015.

ID Prueba	PS-016	
Título	Actualización del riesgo.	
Descripción	Al seleccionar la opción *Recalculate risk* del menú superior del visualizador interactivo se debe mostrar un mensaje con el nuevo riesgo calculado para la IDN.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario crea/carga una IDN de pequeño tamaño para ser visualizada en el visualizador interactivo.
	3	El usuario selecciona la opción *Recalculate risk*.
Resultado esperado	La aplicación muestra el riesgo calculado para la IDN.	

Tabla 7.16: Prueba PS-016.

ID Prueba	PS-017	
Título	Limpieza del visualizador interactivo.	
Descripción	Al seleccionar la opción *Clear* del menú superior del visualizador interactivo se deben eliminar todos los nodos y conexiones de la IDN actual.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario crea/carga una IDN de pequeño tamaño para ser visualizada en el visualizador interactivo.
	3	El usuario selecciona la opción *Clear*.
Resultado esperado	La aplicación borra todos los nodos y conexiones de la IDN.	

Tabla 7.17: Prueba PS-017.

ID Prueba	PS-018	
Título	Uso del zoom del visualizador interactivo.	
Descripción	Al seleccionar las opciones *Zoom in*, *Zoom out* y *Reset zoom* el gráfico del visualizador interactivo debe responder a dichas acciones acercando, alejando o devolviendo a su posición inicial la visualización de la IDN.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario crea/carga una IDN de pequeño tamaño para ser visualizada en el visualizador interactivo.
	3	El usuario selecciona las opciones *Zoom in*, *Zoom out* y *Reset zoom*.
Resultado esperado	La aplicación aleja o acerca (o reinicia) visualmente el gráfico de la IDN.	

Tabla 7.18: Prueba PS-018.

ID Prueba	PS-019	
Título	Guardado de una IDN.	
Descripción	Al seleccionar la opción *Save* del menú superior del visualizador (tanto interactivo como estático), la IDN debe ser guardada como fichero de sistema en la localización elegida por el usuario.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario crea/carga una IDN para ser visualizada.
	3	El usuario selecciona la opción *Save*.
	4	El usuario selecciona el directorio en el que desea guardar la red y el nombre del fichero en el que será almacenada.
Resultado esperado	La IDN es guardada en un fichero con el formato .net en la localización especificada por el usuario.	

Tabla 7.19: Prueba PS-019.

ID Prueba	PS-020	
Título	Exportación de una IDN	
Descripción	Al seleccionar la opción *Export* del menú superior del visualizador (tanto interactivo como estático), la red debe ser exportada a una imagen en la localización elegida por el usuario.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario crea/carga una IDN para ser visualizada.
	3	El usuario selecciona la opción *Export*.
Resultado esperado	La IDN es exportada a una imagen en formato .gif en la carpeta *output* situada en la localización especificada por el usuario para guardar la IDN.	

Tabla 7.20: Prueba PS-020.

ID Prueba	PS-021	
Título	Generación de una solución.	
Descripción	Al seleccionar la opción *Generate solution* del menú superior del visualizador (tanto interactivo como estático) se debe generar una solución para la IDN.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario crea/carga una IDN para ser visualizada.
	3	El usuario selecciona la opción *Generate solution*.
Resultado esperado	La aplicación genera una búsqueda de la solución óptima para la IDN actual y muestra el Frente de Pareto con los resultados en un gráfico.	

Tabla 7.21: Prueba PS-021.

ID Prueba	PS-022	
Título	Selección de un punto de la solución.	
Descripción	Al seleccionar un punto coste-beneficio correspondiente a una solución del gráfico de resultados, ese punto debe ser utilizado por la aplicación para aplicar contramedidas en la IDN y crear una nueva version *secured* (asegurada).	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario crea/carga una IDN de pequeño tamaño para ser visualizada.
	3	El usuario selecciona la opción *Generate solution*.
	4	El usuario selecciona un punto de los disponibles en el gráfico de resultados de la solución.
Resultado esperado	La aplicación crea una nueva IDN asegurada en la carpeta *output* situada en el directorio de la IDN actual.	

Tabla 7.22: Prueba PS-022.

ID Prueba	PS-023	
Título	Verificación de formularios.	
Descripción	Al introducir valores en los distintos formularios de la aplicación se debe realizar una comprobación para verificar que esos valores son válidos. Si no lo son, se muestra un mensaje de error.	
Secuencia	Paso	Acción
	1	El usuario inicia la aplicación.
	2	El usuario introduce datos no válidos en cualquiera de los formularios presentes en la aplicación.
Resultado esperado	La aplicación muestra un mensaje de error.	

Tabla 7.23: Prueba PS-023.

7.1.1. Relación de pruebas superadas

En la *Tabla 7.23* se muestran los resultados obtenidos en las pruebas realizadas a fecha de la finalización del período de pruebas de la aplicación DEFIDNET.

ID Prueba	Superada	No superada
PS-001	X	
PS-002	X	
PS-003	X	
PS-004	X	
PS-005	X	
PS-006	X	
PS-007	X	
PS-008	X	
PS-009	X	
PS-010	X	
PS-011	X	
PS-012	X	
PS-013	X	
PS-014	X	
PS-015	X	
PS-016	X	
PS-017	X	
PS-018	X	
PS-019	X	
PS-020	X	
PS-021	X	
PS-022	X	
PS-023	X	

Tabla 7.24: Relación de pruebas superadas.

7.2. Evaluación del sistema

Tras la realización y superación de las distintas pruebas definidas, se puede concluir que la aplicación cumple con todos los requisitos especificados y funcionalidades necesarias. Por tanto, DEFIDNET está lista para su despliegue y uso en entornos reales.

Capítulo 8

Conclusiones y líneas futuras de investigación

8.1. Conclusiones

Con DEFIDNET se ha conseguido presentar una herramienta eficaz para el diseño de IDNs, siendo capaz de generar soluciones óptimas y alcanzando el mejor coste-beneficio, un paso hacia delante respecto a otras herramientas similares. De esta manera, se suplen necesidades cuya solución no se ha encontrado en los programas ya existentes para la creación de IDNs.

Se considera que DEFIDNET cumple los requisitos especificados al inicio del proyecto de manera satisfactoria. Se han alcanzado todos los objetivos y se ha cubierto la funcionalidad deseada. Por tanto, el proyecto puede ser valorado como exitoso.

El desarrollo no ha sufrido grandes contratiempos debidos a cuestiones internas del propio proyecto. Sin embargo, se produjo un retraso de dos meses ajeno a éste que desembocó en la modificación de la fecha de entrega establecida. Si bien dicho retraso supuso una desviación respecto al plan inicial, una vez la actividad fue retomada, el

flujo de actividad se llevó a cabo según lo previsto.

Se han presentado también problemas durante la fase de codificación debido a la complejidad de algunas operaciones. Sin embargo, éstos se han solucionado en un período corto de tiempo. Algunos de estos problemas se han producido debido a errores de visualización en las imágenes exportadas mediante GraphViz. Sin embargo, se debían a un fallo en la configuración de la herramienta y por tanto ajeno a DEFIDNET. Así mismo, la forma de visualizar las IDN en el visualizador interactivo fue difícil de gestionar debido a problemas de espacio y escalabilidad, que al final se resolvieron haciendo una distinción entre IDNs de pequeño tamaño (con 50 nodos o menos) e IDNs de gran tamaño (el resto), siendo estas últimas exportadas como imagen.

Una vez finalizada su implementación y superadas todas las pruebas para asegurar la calidad del código, DEFIDNET se presenta como una aplicación orientada a usuarios con conocimientos sobre IDNs con una interfaz simple e intuitiva que permite aunar en un único proceso todas las operaciones necesarias para llevar a cabo el diseño de dichas redes de manera efectiva y óptima.

Como parte final, se pretende registrar la aplicación a través de la oficina de registro de la universidad. Dicho software se corresponde con el código completo de DEFIDNET, que incluye la funcionalidad inicial de la que partía este proyecto, y la interfaz y mejoras desarrolladas.

8.2. Líneas futuras de investigación

A continuación se definen líneas futuras de investigación para posibles ampliaciones de la funcionalidad de DEFIDNET que completen el código y, por tanto, desemboquen en una mejora en la experiencia de uso de la herramienta, aportando nuevas características o puliendo las ya existentes.

Estas son las posibles líneas a seguir para completar y mejorar la aplicación desarrollada:

- Añadir funcionalidad para poder editar en el visualizador interactivo de la aplicación nuevas IDNs de características más complejas, como IDN compuestas (de arquitectura *mixed-hc*[1] o *mixed-hr*[2], por ejemplo).
- Del mismo modo, permitir la edición de IDNs de gran tamaño (con un número de nodos superior a 50 nodos) en el visualizador interactivo, modificando la forma en la que se crea la visualización inicial de la IDN para soportar dicho número de nodos consiguiendo un dibujado de la red correcto.
- Permitir la redimensión automática de los gráficos al realizar las operaciones de *zoom in* (acercar) y *zoom out* (alejar) para poder gestionar IDNs más grandes.
- Incluir soporte en otros idiomas, como pueda ser el idioma español.

[1] Arquitectura combinada compuesta de una IDN jerárquica y una IDN centralizada.

[2] Arquitectura combinada compuesta de una IDN jerárquica y una IDN con arquitectura de anillo.

Glosario

- **Ciberespacio** Ámbito de comunicaciones constituido por una red informática.
- **Ciberseguridad** Procedimientos enfocados a la protección de la información almacenada en el ciberespacio.
- ***Cluster*** Conjunto de computadores.
- **DC** *Data Collector*.
- **DDF** *Distributed Detection Function*.
- **ESF** *Event Sharing Function*.
- **Evento discreto** Evento cuyo estado cambia en instantes espaciados en el tiempo.
- **Firma** Patrón de ataque.
- **Frente de Pareto** Conjunto de valores óptimos de una función objetivo de tal manera que representan el punto de equilibro en el que ningún elemento de la función puede mejorar su situación actual sin perjudicar al otro.
- **HIDS** *Host* IDS.
- ***Host*** Computador conectado a una red.

- **IDE** *Integrated Development Environment.*
- **IDMsg** *Intrusion Detection Message.*
- **IDN** *Intrusion Detection Network.*
- **IDS** *Intrusion Detection System.*
- **IIDM** *Input Intrusion Detection Messages.*
- **LD** *Local Detection.*
- **LDA** *Local Detection and Alert Sharing.*
- **LDF** *Local Detection Function.*
- **LE** *Local Events.*
- ***Mixed-hc*** Arquitectura combinada compuesta por una IDN jerárquica y una IDN centralizada.
- ***Mixed-hr*** Arquitectura combinada compuesta de una IDN jerárquica y una IDN con arquitectura de anillo.
- **MVC** *Model-View-Controller.*
- **NIDS** *Network* IDS.
- **OIDM** *Output Intrusion Detection Messages.*
- **PC** *Pure Correlation.*
- **RA** *Responsive Actions.*
- **RC** *Remote Correlation.*
- **RCD** *Remote Correlation and Detection.*
- **RF** *Responsive Function.*

- **SPEA** *The Strength Pareto Evolutionary Algorithm.*
- **TFG** Trabajo de Fin de Grado.

Bibliografía

[1] Amer, S. y Hamilton, J. *Intrusion Detection Systems (IDS) taxonomy. A short review.* Journal of Software Technology, Vol 13 (2010).

[2] Corona, I., Giacinto, G. y Roli, F. *Adversarial Attacks Against Intrusion Detection Systems: Taxonomy, Solutions and Open Issues.* Information sciences, Vol 239 (2013). 201-225.

[3] Fonseca, C. y Fleming, P. *Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization.* Proceedings of the 5th International Conference on Genetic Algorithms (1993), 416-423.

[4] Fung, C. *Collaborative Intrusion Detection Networks and Insider Attacks.* Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, Vol 2 (2013). 63-74.

[5] Fung, C. y Boutaba, R. *Design and management of collaborative intrusion detection networks.* The 15th IFIP/IEEE International Symposium on Integrated Network Management (2013). 955-961.

[6] Liao, H., Lin, C. y Tung, K. *Intrusion Detection System: A comprehensive review.* Journal of Network and Computer Applications, Vol 36 (2013). 1, 16-24.

[7] Pastrana, S. *Attacks Against Intrusion Detection Networks: Evasion, Reverse Engineering and Optimal Countermeasures.* PhD. Carlos III University, 2014.

[8] Ten, C., Manimaran, G. y Liu, C. *Cybersecurity for Critical Infrastructures: Attack and Defense Modeling.* IEEE Transactions on systems, man, and cybernetics PART A: systems and humans, Vol 40 (2010). 853-865.

[9] Wei, H., Frinke, D., Carter, O. y Ritter, C. *Cost-Benefit Analysis for Network Intrusion Detection Systems.* CSI 28th Annual Computer Security Conference (2001).

[10] Xenakis, C., Panos, C. y Stavrakakis, I. *A comparative evaluation of intrusion detection architectures for mobile ad hoc networks.* Computers & Security, Vol 30 (2011). 63-80.

[11] Zhou, C., Leckie, C. y Karunasekera, S. *A survey of coordinated attacks and collaborative intrusion detection.* Computers & Security, Vol 29 (2010). 124-140.

[12] Zitzler, E., Laumanns, M. y Thiele, Lothar. *SPEA2: Improving the Strength Pareto Evolutionary Algorithm.* TIK-Report, Vol 103 (2001).

[13] Zitzler, E. y Thiele, Lothar. *Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Evolutionary Algorithm.* IEEE Transactions on Evolutionary Computation, Vol 3(4) (1999). 257-271.

[14] Agile Modeling. *UML 2 Use Case Diagrams: An Agile Introduction.*
http://www.agilemodeling.com/artifacts/useCaseDiagram.htm

[15] Departamento de ingeniería y tecnología de computadores. Universidad de Murcia. *Seguridad y protección.*
http://www.ditec.um.es/so/apuntes/teoria/tema7.pdf

[16] Department of Computer Science (George Mason University). *ECJ22 Project.*
http://cs.gmu.edu/~eclab/projects/ecj/

[17] Dr. John B. Matthews website. *GraphPanel.*
https://sites.google.com/site/drjohnbmatthews/graphpanel

[18] GraphViz Documentation. *Graphviz and Dynagraph. Static and Dynamic Graph Drawing Tools.*

http://www.graphviz.org/Documentation/EGKNW03.pdf

[19] Information Sciences Institute, USC. *The Network Simulator NS-2.*

http://www.isi.edu/nsnam/ns/

[20] JfreeChart website. *JFreeChart API Documentation.*

http://www.jfree.org/jfreechart/api/javadoc/index.html

[21] Laszlo Szathmary. *GraphViz Java API.*

http://www.loria.fr/~szathmar/off/projects/java/GraphVizAPI/index.php

[22] Marco de desarrollo de la junta de Andalucía. *Especificación de requisitos de sistema.*

http://goo.gl/QfkPr5

[23] Methods and Tools. *Understanding Use Case Modeling.*

http://www.methodsandtools.com/archive/archive.php?id=24

[24] Microsoft MSDN. *Model-View-Controller.*

http://msdn.microsoft.com/en-us/library/ff649643.aspx

[25] Ministerio de Empleo y Seguridad Social. *Bases y tipos de cotización 2014.*

http://goo.gl/xlxRgF

[26] MIT. *Red Hat Enterprise Linux 4: Manual de seguridad.*

http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-sg-es-4/ch-detection.html

[27] NetSecurity. *Introduction to Intrusion Detection Systems (IDS).*

http://netsecurity.about.com/cs/hackertools/a/aa030504.htm

[28] OMNeT++ website. *OMNeT++.*

http://www.omnetpp.org/home/what-is-omnet

[29] Oracle. *The Java Tutorials.*

http://docs.oracle.com/javase/tutorial/

[30] Ponemon Institute. *2013 Cost of Cyber Crime Study: Global Report.*

http://goo.gl/r7BH6Y

[31] Scalable Networks. *GloMoSim: A Library for Parallel Simulation of Large-scale Wireless Networks.*

http://www.scalable-networks.com/pdf/glomosim.pdf

[32] TechRepublic. *Understanding how an Intrusion Detection System (IDS) works.*

http://http://goo.gl/eiD3ze

[33] Toronto Users Group, Vol 12. *Java: Write once, run everywhere.*

http://www.tug.ca/articles/Volume12/V12N4/V12N4_Javier_Java.html

[34] University of Washigton. *Software Project Estimation.*

http://goo.gl/jtq7La

Anexo I. Manual de usuario

1. Descripción de la aplicación

DEFIDNET es una aplicación destinada al diseño y análisis de redes de detección de intrusiones, o IDNs por sus siglas en inglés (*Intrusion Detection Networks*).

En este manual de usuario se encuentran las distintas directrices e instrucciones necesarias para hacer un uso correcto y eficaz de la herramienta desarrollada.

DEFIDNET permite la simulación de IDNs y su posterior edición, proporcionando el siguiente repertorio de operaciones:

- Creación de IDNs.
- Edición de IDNs.
- Carga de IDNs.
- Unión de IDNs.
- Visualización estática de IDNs.
- Visualización interactiva de IDNs.
- Guardado de IDNs.
- Exportación a imagen de IDNs.
- Generación de soluciones óptimas para IDNs.

2. Requisitos

Requisitos mínimos del sistema:

- Java 1.7 o superior.
- GraphViz 2.38 o superior.

Es posible utilizar varias funcionalidades de DEFIDNET sin GraphViz. Sin embargo, se recomienda encarecidamente su instalación en el sistema, puesto que tanto la exportación a imagen (*11.2. Exportación de la IDN*) de una IDN como el proceso de generación de una solución (*11.4. Generación de soluciones óptimas*) requieren de la herramienta.

Por tanto, para disponer de la funcionalidad de DEFIDNET por completo es necesario obtener GraphViz 2.38 o superior.

Enlace a la página del proyecto GraphViz: www.graphviz.org/

3. Preparación de DEFIDNET

Para poder hacer uso de DEFIDNET el usuario debe descargar el fichero JAR de la aplicación en la siguiente URL y ejecutarlo:

www.seg.inf.uc3m.es/~spastran/DEFIDNET.jar

4. Fichero de configuración

El fichero de configuración contiene información sobre la localización de la herramienta GraphViz y sobre la carpeta temporal que se debe usar en el proceso de exportación de una IDN a imagen.

En el caso de que el usuario utilice la aplicación por primera vez o de que el fichero no sea encontrado, se pedirá mediante un formulario la inserción de la siguiente información:

- ***GraphViz dot***. La ruta absoluta del ejecutable dot. En Windows dicho ejecutable se encuentra usualmente en el subdirectorio *release/bin/* de la carpeta de la herramienta GraphViz, mientras que en sistemas operativos Linux puede encontrarse en */usr/bin/dot*.
- ***Temporal directory***. La ruta absoluta a la carpeta temporal que se desea utilizar. Debe tener permisos de lectura y escritura.

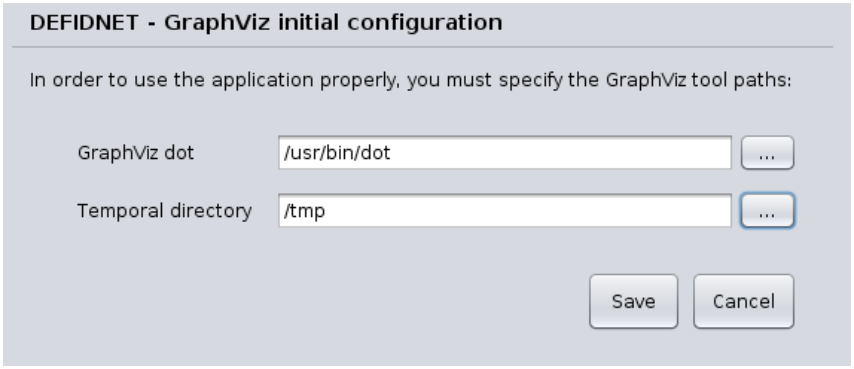

Figura 8.1: Formulario de configuración inicial.

Una vez introducidos los parámetros se debe pulsar el botón *Save*. DEFIDNET almacenará la información en el fichero app.config.

Para modificar el fichero app.config, el usuario puede presionar el botón de configuración con forma de ruedas dentadas del menú principal. Un formulario idéntico al anterior aparecerá, y se podrán modificar las rutas.

Si se desea crear/modificar el fichero de configuración de manera manual se ha de acceder a la carpeta raíz del proyecto DEFIDNET y generar o editar éste mediante un editor de texto. Se debe seguir el formato mostrado en la figura 8.2.

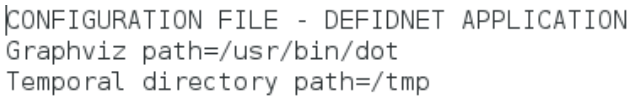

Figura 8.2: Fichero de configuración.

El usuario únicamente debe modificar las rutas, cualquier otra modificación en el formato del fichero hará fallar la aplicación.

5. Menú principal

El menú principal de DEFIDNET dispone de las tres principales opciones que el usuario puede elegir al inicio del proceso:

- *New IDN*. Permite la creación de una nueva IDN.
- *Load IDN*. Permite la carga en la aplicación de una IDN ya existente almacenada en formato .net.
- *Join IDNs*. Permite combinar dos o más IDNs en una única IDN.

Además, contiene el botón de modificación del fichero de configuración de GraphViz descrito en el apartado anterior (denotado por la imagen de ruedas dentadas).

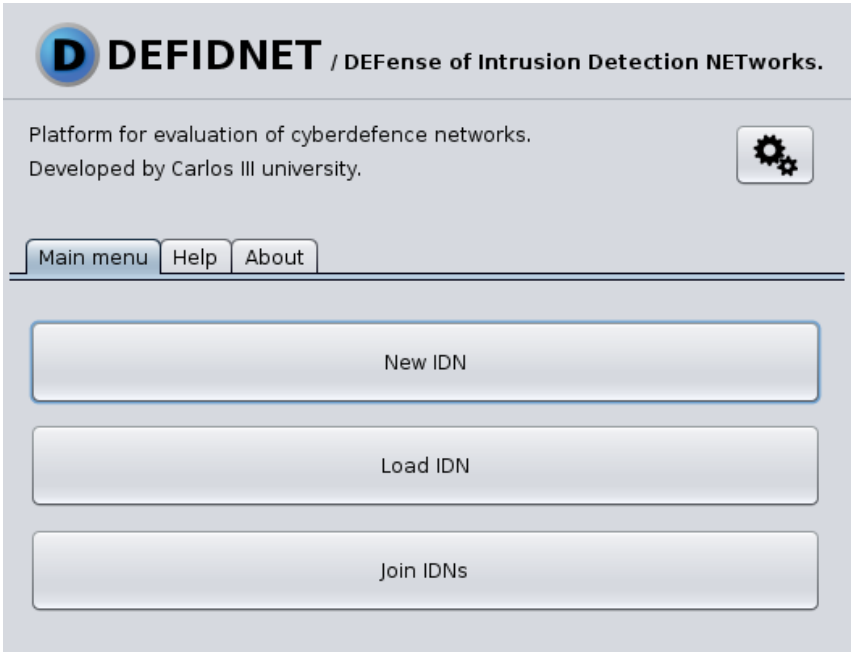

Figura 8.3: Menú principal de la aplicación.

6. Creación de una nueva IDN

Al seleccionar la opción *New IDN* en el menú principal de la aplicación el usuario comienza un proceso de introducción de datos en distintos formularios para la creación de una nueva IDN.

En el primer formulario, *New IDN Form*, el usuario deberá introducir información global relativa a la nueva IDN en su conjunto:

- ***Network architecture***. Describe la arquitectura que tendrá la IDN. Es un paso importante, puesto que la disposición visual de los nodos, así como los roles creados dependen de la arquitectura elegida. Se puede elegir entre: *centralized* (arquitectura centralizada), *hierarchical* (arquitectura jerárquica), *fully-distributed* (arquitectura distribuida), *partially-distributed* (arquitectura parcialmente distribuida) y *ring* (arquitectura en forma de anillo). Para leer

más información acerca de los distintos tipos de arquitectura, véase el punto *2.3.3. Arquitecturas de IDNs.*

- ***Influence distribution***. Especifica la forma de distribución de influencias entre nodos. Las distintas opciones son: *Uniform* (distribución uniforme de influencias), *fractional* (distribución fraccional), *exponential* (distribución exponencial) y *gaussian* (distribución gaussiana).

- ***Number of nodes***. Define el número total de nodos de la IDN. Si el número total de nodos es menor menor o igual a 50, la IDN especificada más tarde será mostrada en el visualizador interactivo de la aplicación (*10. Visualizador interactivo*); en caso contrario, la IDN será mostrada en el visualizador estático (*9. Visualizador estático*).

- ***Connection probability***. Sólo en redes parcialmente distribuidas. Define el porcentaje de conexión que cada nodo tiene con el resto. Por ejemplo, en una IDN parcialmente distribuida con 10 nodos y una probabilidad de conexión del 80 %, cada nodo estará conectado a un máximo de otros 8 nodos de la IDN.

- ***Number of levels***. Sólo en redes jerárquicas. Define el número de niveles de los que se compondrá el árbol jerárquico generado.

- ***Number of nodes per level***. Sólo en redes jerárquicas. Define el número de hijos para cada nodo. No es posible especificar un número distinto de hijos para cada nodo.

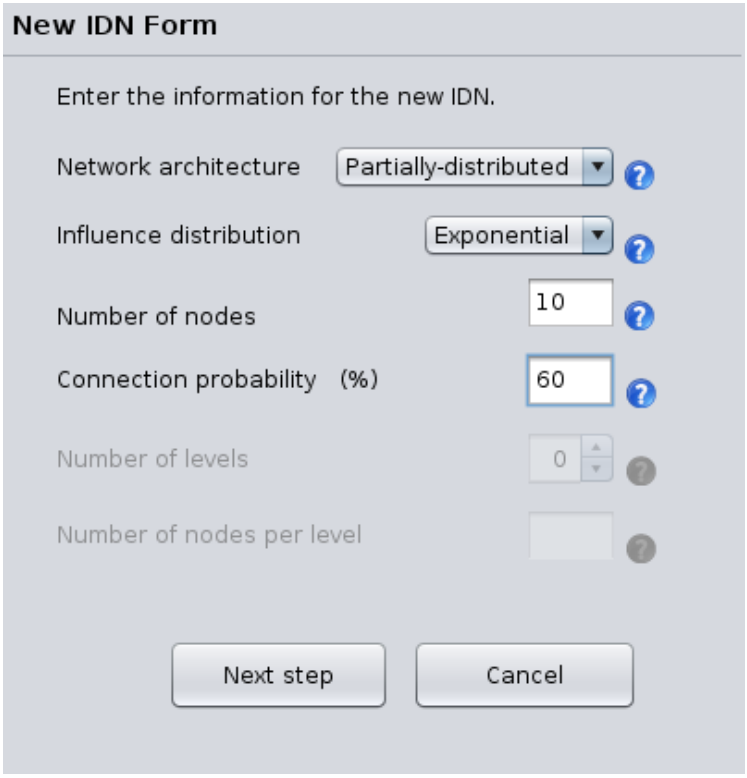

Figura 8.4: Formulario de creación de una IDN.

En el siguiente o siguientes formularios se introduce información relativa a los distintos campos que conciernen a un determinado rol. De tal manera que si la IDN consta de un único tipo de nodo sólo aparecerá un formulario y si la IDN consta de tres roles distintos aparecerán tres formularios.

Los datos a rellenar por el usuario son:

- ***Probability of being attacked***. Define la probabilidad de que los nodos con el rol configurado sean víctima de un ataque.
- ***Percentage of nodes attacked***. Porcentaje de nodos atacados de dicho rol en caso de que se produzca el ataque.
- ***Budget***. Presupuesto del que se dispone para proteger la IDN. El presupuesto

se expresa en unidades, que pueden ser monetarias, de recursos humanos, etcétera.

- ***Costs distribution***. Distribución de los costes para aplicar contramedidas de protección a una IDN. Existen las siguientes opciones: *Uniform* (distribución uniforme de influencias), *fractional* (distribución fraccional), *exponential* (distribución exponencial) y *gaussian* (distribución gaussiana).

- ***Impacts of each attack***. Impactos sobre los nodos con el rol configurado que producen los distintos tipos de ataque (véase el punto *2.3.4. Tipos de ataque*).

Las características especificadas para un rol se aplican a todos los nodos pertenecientes a ese rol. Más adelante, es posible modificar individualmente cada nodo.

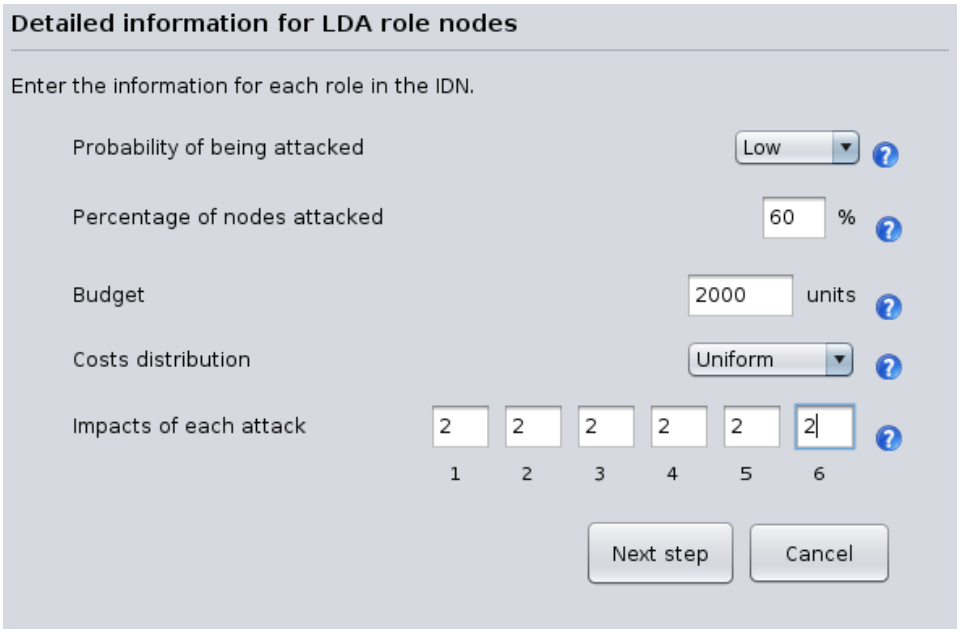

Figura 8.5: Formulario de creación por rol detallado de una IDN.

Tras rellenar los formularios especificados, la IDN creada será visualizada. Dependiendo de su tamaño se utilizará el visualizador estático (más de 50 nodos) o el interactivo.

En este nuevo paso, el usuario tiene la posibilidad de realizar nuevas operaciones sobre la IDN. Véase la sección *11. Operaciones sobre la IDN actual*.

Así mismo, se recomienda leer los puntos referentes a los visualizadores estático e interactivo (secciones *9. Visualizador estático* y *10. Visualizador interactivo*, respectivamente).

7. Carga de una IDN existente

Para cargar una IDN almacenada en un fichero de sistema, el usuario debe pulsar la opción *Load IDN* del menú principal.

El usuario deberá seleccionar en el navegador de ficheros el archivo de extensión .net que contiene la IDN deseada.

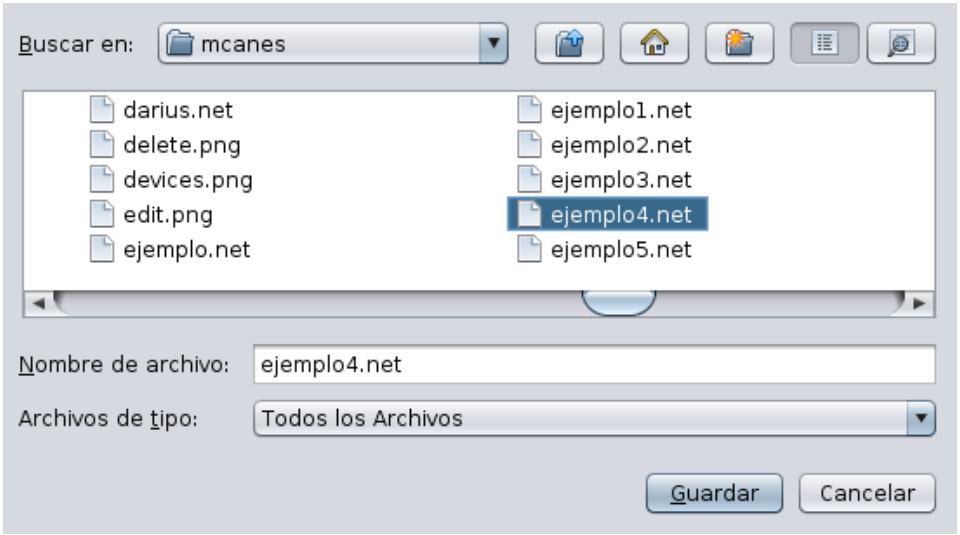

Figura 8.6: Visor de ficheros.

Una vez cargada la IDN en la aplicación, ésta será mostrada al usuario mediante el visualizador estático o el visualizador interactivo (dependiendo de sus dimensiones) y el usuario podrá proceder a operar con ella.

8. Combinación de IDNs

La combinación o unión de IDNs se lleva a cabo al pulsar el botón *Join IDNs*.

El usuario deberá pulsar el botón *Add IDN* del formulario *Join IDNs Form* y seleccionar el fichero .net de la IDN que desea combinar con otras. Debe repetir este proceso para añadir el resto de IDNs a combinar.

En el listado *Added IDNs* se pueden visualizar las IDNs ya seleccionadas.

Cuando el usuario no desee añadir más IDNs, deberá pulsar el botón *Continue*.

Figura 8.7: Formulario de combinación de distintas IDNs.

Aparecerá un formulario para cada IDN seleccionada donde el usuario podrá seleccionar los nodos que se utilizarán como unión. Éstos, se unirán formando una arquitectura totalmente distribuida. Como mínimo, el usuario debe seleccionar un nodo para cada IDN.

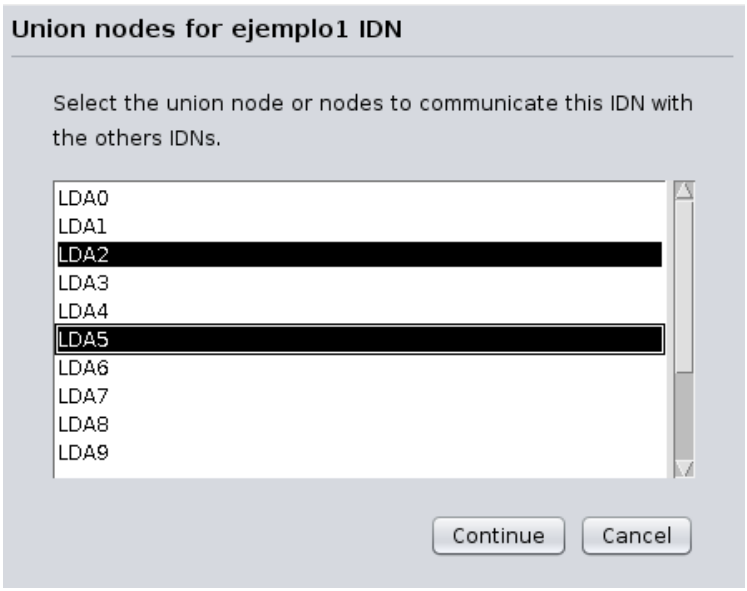

Figura 8.8: Formulario de nodos de unión.

Cuando se hayan rellenado todos los formularios, aparecerá un selector de ficheros para que el usuario pueda especificar donde desea almacenar la IDN compuesta y el nombre que desea otorgarle. El fichero .net final tendrá el formato XXX_joined.net, donde XXX representa el nombre elegido por el usuario. Así mismo, se creará una carpeta *XXX_joined_ouput* en el directorio elegido donde se almacenarán el .dot (formato propio de GraphViz) y la imagen de la IDN en formato .gif.

9. Visualizador estático

El visualizador estático no permite la visualización de IDNs dentro de DEFIDNET, sino que éstas son exportadas a formato imagen (siguiendo el mismo proceso especificado en el punto *11.2. Exportación de la IDN*). Las IDNs de un

tamaño superior a 50 nodos son derivadas a este visualizador puesto que no pueden ser dibujadas en el formato interactivo por motivos de usabilidad y escalabilidad.

El visualizador estático no puede mostrar las IDNs mediante grafos y dispone de opciones más reducidas. Éste, permite el guardado de la IDN, su exportación y la generación de una solución óptima. Sin embargo, no permite la edición de la IDN (inserción de nuevos nodos, modificación de conexiones, etcétera).

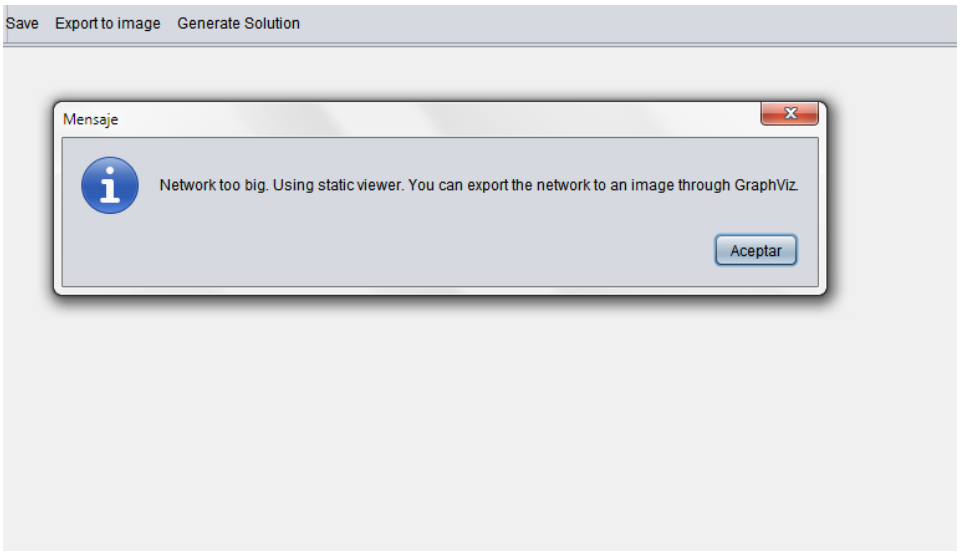

Figura 8.9: Visualizador estático de DEFIDNET.

El usuario puede realizar todas las operaciones posibles de manera manual editando los ficheros de sistema. Véase la sección *12. DEFIDNET en modo manual*.

10. Visualizador interactivo

El visualizador interactivo de DEFIDNET es utilizado para visualizar y editar IDNs de un tamaño inferior o igual a 50 nodos.

Permite realizar las siguientes opciones:

- Visualización mediante figuras interactivas.

- Inserción de nuevos nodos.
- Modificación de los nodos existentes.
- Eliminación de los nodos.
- Creación, modificación y eliminación de conexiones entre nodos.
- Guardado de la IDN.
- Exportación a imagen de la IDN.
- Generación de soluciones óptimas para la IDN.

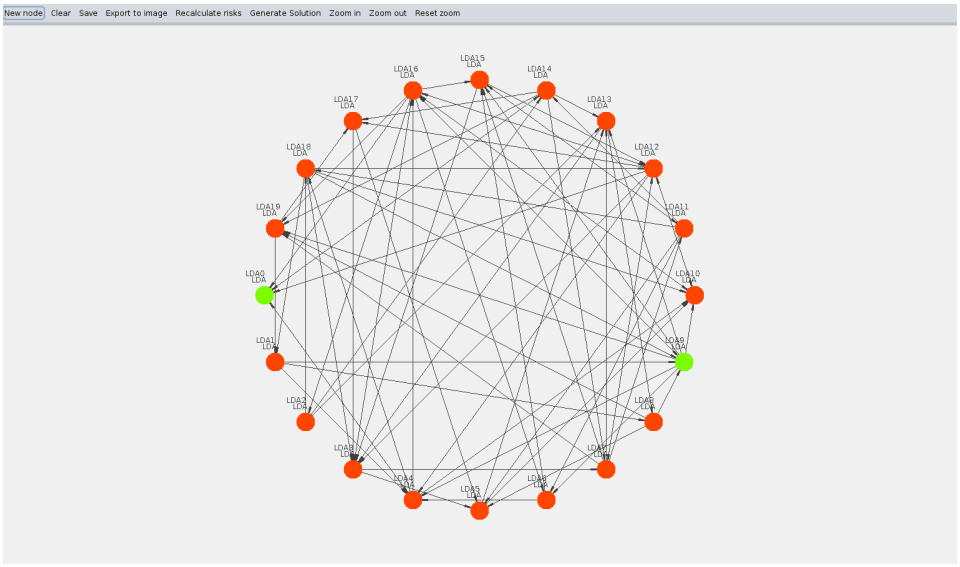

Figura 8.10: Visualizador interactivo de DEFIDNET.

El visualizador interactivo se compone de un menú superior y de una zona de dibujo en la que se visualizan las distintas figuras que representan los nodos y sus conexiones.

En las siguientes subsecciones se describen las características de este visualizador.

10.1. Figuras

Las figuras utilizadas en la zona de dibujo del visualizador interactivo son:

- **Círculo.** Los círculos simbolizan los distintos nodos de una IDN. Sobre cada círculo hay una leyenda con el nombre del nodo y su rol. El color de un círculo define las probabilidades de ser atacado que tiene el nodo que representa.
- **Línea dirigida.** Cada línea dirigida une dos círculos distintos. Ésta, representa la conexión existente entre dos nodos. Su dirección describe el sentido en el que se traspasa la influencia en la relación, de tal forma que si se tiene una flecha dirigida con origen en el círculo A y destino en el círculo B, el nodo A está influenciando al nodo B (es decir, traspasando información con un determinado peso).

10.2. Código de colores

Los distintos nodos que forman una IDN tienen distintas probabilidades de ser atacados. Esta probabilidad se muestra gráficamente mediante el color del círculo que representa al nodo.

Siguiendo una carta de colores predefinidos, los tonos fríos tienden a indicar una menor probabilidad de ser atacado (siendo el verde indicativo de una probabilidad prácticamente nula o nula) y los tonos cálidos tienden a indicar una mayor probabilidad de ser atacado (siendo el rojo el nivel más alto). Los valores van del 0 al 1, equivaliendo 0 al 0 % de riesgo y el 1 al 100 %.

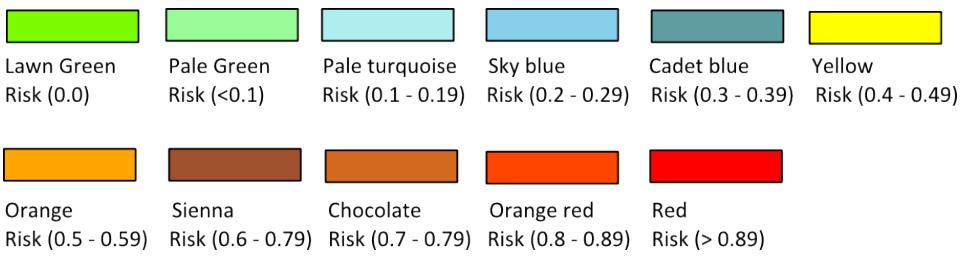

Figura 8.11: Carta de colores para medir la probabilidad de ataque a un nodo.

10.3. Selección

Para seleccionar un círculo (y por lo tanto, un nodo) el usuario simplemente debe pulsar el botón izquierdo de su ratón sobre el círculo deseado. Deberá aparecer una línea delineante roja alrededor del círculo.

Para eliminar la selección de un círculo debe pulsarse el botón izquierdo del ratón fuera de su área. La línea delineante roja desaparecerá indicando que el círculo ya no está seleccionado.

10.4. Selección múltiple

La selección múltiple permite seleccionar más de un nodo al mismo tiempo.

Para mantener la selección anterior y añadir un nuevo círculo seleccionado, el usuario deberá mantener pulsada la tecla *shift* mientras realizar la operación normal de selección. Sino, el nuevo nodo quedará seleccionado, pero el anterior perderá su selección.

Por tanto, se debe usar la combinación *shift + click izquierdo* para utilizar la selección múltiple.

Al igual que con la selección individual, aparecerá una línea delineante roja alrededor del círculo para indicar que éste ha quedado seleccionado correctamente.

10.5. Limpieza del visualizador

El usuario puede borrar todos los nodos y conexiones existentes en una IDN presionando el botón *Clear*.

Esta acción permite al usuario insertar nodos en una IDN desde cero.

10.6. Zoom

El usuario puede utilizar los controles *Zoom in* y *Zoom out* para acercar y alejar el grafo, respectivamente.

11. Operaciones sobre la IDN actual

En el visualizador de IDNs el usuario puede seleccionar en el menú de la parte superior distintas opciones a realizar. Este menú es común a los dos tipos de visualizadores existentes (si bien el estático carece de las opciones referidas al entorno visual) y contiene las operaciones globales sobre la IDN en general.

Mientras, el menú flotante (click derecho sobre una figura) asociado a cada uno de los nodos de la IDN está reservado únicamente para el visualizador interactivo y contiene operaciones especificas para el nodo individual.

En los siguientes apartados se especifican la información detallada y los pasos necesarios de cada una de las operaciones sobre una IDN existente.

11.1. Guardado de la IDN

La opción de guardado permite al usuario almacenar de manera permanente las IDNs con las que está trabajando en formato .net. Dicho formato es el utilizado por DEFIDNET para guardar y posteriormente cargar IDNs.

Al seleccionar la opción *Save* del menú superior del visualizador (tanto estático como interactivo) aparecerá un selector de ficheros que permitirá al usuario guardar la IDN en la ruta deseada.

Si el nombre del fichero especificado ya existe se mostrará un mensaje de alerta preguntando al usuario si desea sobreescribirlo.

11.2. Exportación de la IDN

Esta función permite exportar IDNs a imágenes mediante la herramienta GraphViz. La extensión de las imágenes exportadas es .gif.

Cuando el usuario selecciona la opción *Export to image* pueden darse dos situaciones. Si éste ha guardado previamente la IDN actual mediante la operación *Save*, DEFIDNET recordará la ruta y creará una carpeta (en el caso de que no exista) con el nombre *XXX_output* en ese directorio, donde XXX es el nombre con el que se guardó la IDN. Después, generará los ficheros correspondientes.

Si por el contrario, el usuario no guardó la IDN con anterioridad, deberá seleccionar el nombre y directorio deseados en el gestor de ficheros para realizar esa operación. DEFIDNET realizará el proceso de guardado y generará los ficheros correspondientes al proceso de exportación en la subcarpeta *XXX_output*.

La exportación a imagen genera dos ficheros:

- Un fichero de extensión .dot, utilizado por GraphViz para generar la imagen y que puede ser editado manualmente para modificar el grafo resultante.
- Una imagen .gif con el grafo final generado.

El usuario deberá esperar unos segundos, dependiendo del tamaño de la IDN, para poder visualizar la imagen resultante, que se abrirá automáticamente a través del visor de imágenes predefinido en el equipo utilizado.

Si la imagen no se abriese automáticamente, el usuario puede acceder a la localización de los ficheros resultantes de la exportación y abrirla manualmente.

11.3. Actualización del riesgo de la IDN

La opción *Recalculate risks* actualiza el riesgo total generado de la IDN. El riesgo varía cada vez que se modifica un nodo o se crea una conexión, por ejemplo.

Este proceso no es automático, por lo que el usuario debe pulsar el botón *Recalculate risks* siempre que edite las características de la IDN actual. No es necesario que realice esta operación cuando únicamente modifica el aspecto visual (desplazamiento de figuras).

La aplicación mostrará un mensaje informativo con el nuevo riesgo total de la IDN.

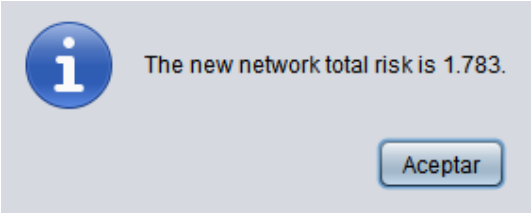

Figura 8.12: Mensaje de aviso con el riesgo total de la IDN.

11.4. Generación de soluciones óptimas

La generación de soluciones óptimas permite analizar una IDN y definir cuál es la mejor solución para protegerla en base a la relación coste-beneficio (siendo beneficio el riesgo mitigado).

Para proceder con esta operación, el usuario debe seleccionar la opción *Generate solution* del menú superior.

Si el usuario previamente guardó la IDN actual, los ficheros resultantes de la generación serán almacenados en el subdirectorio *XXX_output*, donde XXX es el nombre con el que la IDN fue guardada. En el caso contrario, aparecerá un selector de ficheros y el usuario deberá seleccionar el directorio de salida.

A continuación, DEFIDNET guardará la IDN en formato .net y procederá a la generación de la solución. Este proceso puede llevar desde unos pocos segundos a varios minutos, dependiendo del tamaño de la IDN y el número de generaciones especificado. Se crearán varios ficheros con información necesaria para la aplicación que serán almacenados en el directorio *XXX_output*. Durante la espera, es importante que el usuario no modifique ninguno de estos ficheros, pues el proceso podría fallar.

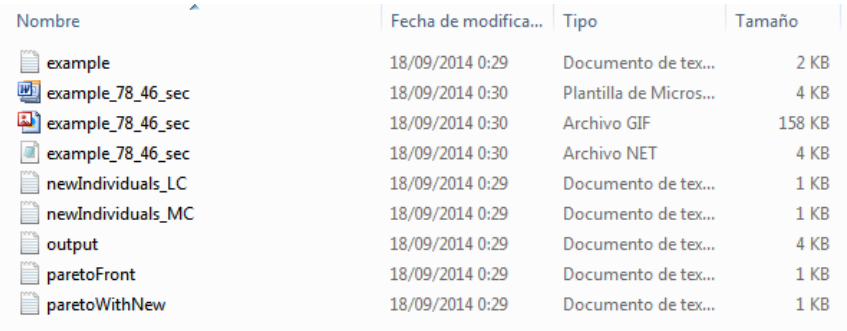

Figura 8.13: Ficheros generados por la generación de una solución.

Tras la generación del Frente de Pareto, la aplicación mostrará un gráfico con los resultados y las posibles soluciones. El usuario deberá hacer doble click en el punto deseado y DEFIDNET creará una nueva IDN asegurada con las contramedidas aplicadas en base al punto de coste-beneficio seleccionado.

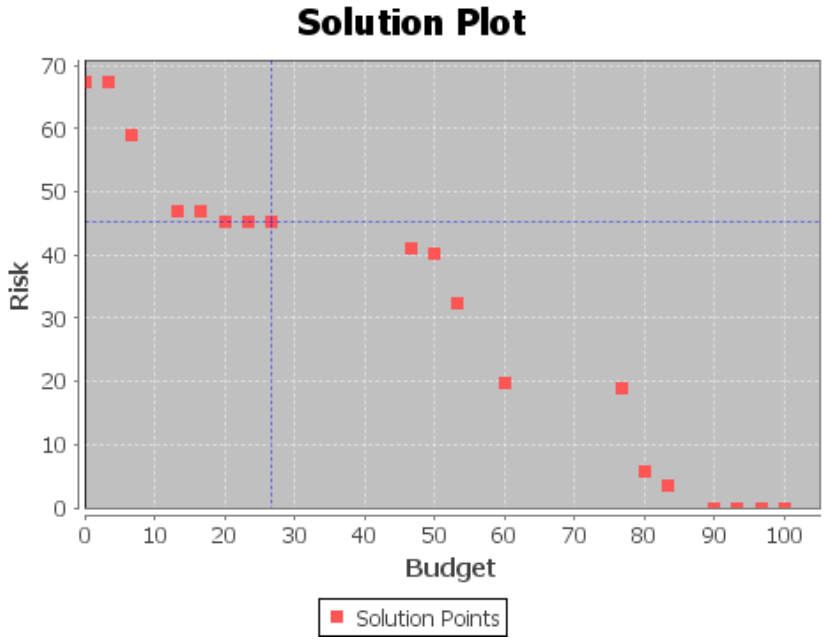

Figura 8.14: Gráfico de resultados de la generación de una solución.

La red generada será almacenada el directorio *XXX_output* especificado anteriormente. Se guardará con el nombre XXX_y_z_sec.net, siendo XXX el nombre de la red original y los valores y y z las coordenadas del punto seleccionado (simbolizando los valores del riesgo mitigado y del presupuesto, respectivamente).

11.5. Estructura de los nodos

Las IDNs están formadas por nodos que analizan el tráfico entrante para detectar patrones de posibles ataques y conexiones entre esos nodos para que éstos puedan comunicarse y traspasar información entre ellos, consiguiendo así un visión más global mediante la correlación de información.

Los nodos de una IDN se componen de los siguientes principales atributos:

- ***Node name***. Nombre identificativo del nodo.
- ***Node type***. Rol del nodo.

- ***Impacts***. Impactos que producen los distintos tipos de ataque (véase el punto *2.3.4 Tipos de ataque*) sobre el nodo individual.
- ***Probabilities***. Probabilidades de ser atacado en cada uno de sus canales por cada unos de los distintos tipos de acciones intrusivas (interrupción, intercepción, modificación y fabricación). Más información en el punto *2.3.4 Tipos de ataque*.
- ***Costs***. Costes de proteger cada uno de sus canales para cada tipo de acción intrusiva.

Se puede encontrar más información sobre las características de un nodo en el punto *2.3.2 Nodos*.

11.6. Creación de un nuevo nodo

Al seleccionar la opción *New node*, la aplicación mostrará un formulario con los datos a introducir para la creación de un nuevo nodo.

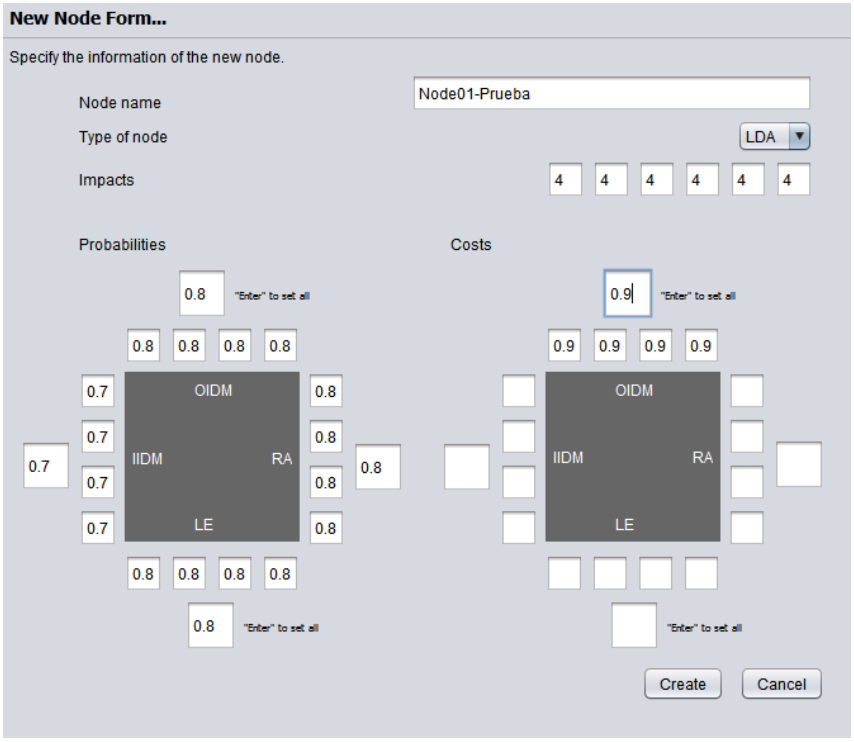

Figura 8.15: Formulario para la creación de un nuevo nodo.

El usuario puede fijar la misma probabilidad o coste para todos los tipos de acciones intrusivas existentes en un canal utilizando los recuadros de mayor tamaño y presionando la tecla *Enter*.

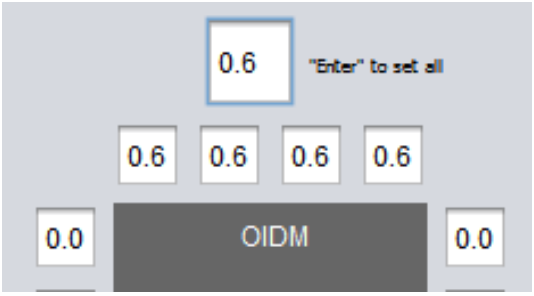

Figura 8.16: Campo para fijar un mismo valor múltiples veces.

11.7. Detalles de un nodo existente

Al seleccionar la opción *Node details* del menú flotante asociado a un nodo (click derecho sobre éste), la aplicación mostrará un formulario con toda su información. Esta ventana únicamente tiene carácter informativo.

11.8. Modificación de un nodo existente

Al seleccionar la opción *Edit node* del menú flotante, la aplicación mostrará un formulario con los datos a introducir para la modificación de un nodo existente.

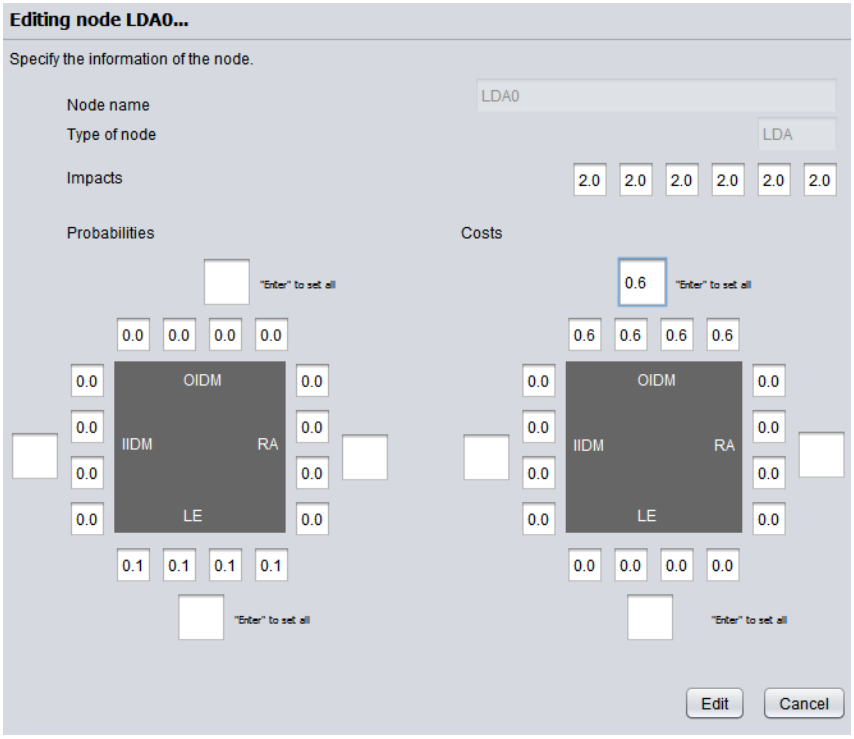

Figura 8.17: Formulario para la modificación de un nodo.

Al igual que en el punto *11.6. Creación de un nuevo nodo*, se puede fijar la misma probabilidad o coste para todos los tipos de acciones intrusivas existentes introduciendo un único valor en el recuadro de mayor tamaño y pulsando la tecla *Enter*.

11.9. Eliminación de un nodo

Eliminar un nodo de la IDN es un tarea muy sencilla. El usuario debe seleccionar la opción *Delete* del menú flotante asociado al nodo (click derecho sobre éste).

El nodo será borrado y todas sus conexiones eliminadas.

11.10. Operaciones sobre conexiones

Las conexiones entre distintos nodos se visualizan mediante líneas dirigidas que los unen. Una línea dirigida tiene un origen y un fin, siendo este último denotado por una forma de triángulo.

Una conexión entre dos nodos puede ser creada (si no existe previamente), eliminada o modificada.

11.10.1. Crear conexión entre nodos

Si entre dos nodos no existe una conexión, se puede crear una nueva siguiendo los pasos enumerados a continuación:

- Se selecciona el primer nodo. Este nodo será el que traspase su influencia al segundo. Es decir, la punta de la flecha dirigida de la figura de la conexión estará en el otro nodo.
- Se selecciona el segundo nodo mediante la selección múltiple (*shift + click izquierdo*). Este nodo recibirá la influencia del primero; es decir, en este nodo se encontrará la cabeza de la flecha.
- Se hace click derecho sobre cualquiera de los dos nodos seleccionados y en el menú flotante se selecciona la opción *Connect*.
- De esta manera, se creará una conexión entre los dos nodos especificados.

Figura 8.18: Conexión entre dos nodos.

11.10.2. Modificar influencia de conexión entre nodos

Para modificar la influencia de una conexión entre dos nodos se deben seguir los siguientes pasos:

- Se selecciona el primer nodo. Este nodo deberá ser el que esté traspasando la influencia que queremos modificar al otro nodo.
- Se selecciona el segundo nodo mediante la selección múltiple (*shift + click izquierdo*).
- Se hace click derecho sobre cualquiera de los dos nodos seleccionados y en el menú flotante se selecciona la opción *Change influece*.
- Aparecerá una pequeña ventana modal con el valor de la influencia de conexión actual que el primer nodo seleccionado está traspasando al segundo. El usuario deberá modificar el valor mencionado y seleccionar la opción *Set influence*.
- De esta manera, la nueva influencia de conexión será fijada.

Figura 8.19: Formulario de modificación de conexión.

11.10.3. Eliminar conexión entre nodos

Esta opción es sumamente similar a las mencionadas anteriormente para operar sobre conexiones.

Si el usuario desea eliminar una conexión entre dos nodos, deberá realizar los siguientes pasos:

- Se selecciona el nodo desde el cual se desea borrar la influencia que traspasa al otro.
- Se selecciona el segundo nodo mediante la selección múltiple (*shift + click izquierdo*).
- Se hace click derecho sobre cualquiera de los dos nodos seleccionados y en el menú flotante se selecciona la opción *Disconnect*.
- De esta manera, la conexión del primer nodo al segundo será eliminada.

12. DEFIDNET en modo manual

Es posible gestionar las IDNs y operar con ellas de manera manual. El usuario deberá acceder a los ficheros de sistema y editarlos.

Dado que el objetivo de DEFIDNET es proveer una interfaz visual para evitar estas acciones y hacer más sencillo el proceso, sólo se reseñan los ficheros más relevantes.

- Fichero de configuración app.config. Se pueden modificar las rutas del fichero de configuración de GraphViz accediendo a él directamente en la carpeta raíz de DEFIDNET. Se puede obtener más información en el punto *8.2 4. Fichero de configuración* de este manual.

- Ficheros .net. Toda la información relativa a una IDN se almacena en su fichero .net correspondiente. El usuario puede operar con los nodos, conexiones, impactos, probabilidades y costes directamente. También puede modificar la arquitectura de la IDN. En la figura 8.20 se puede ver un extracto de uno de estos ficheros.

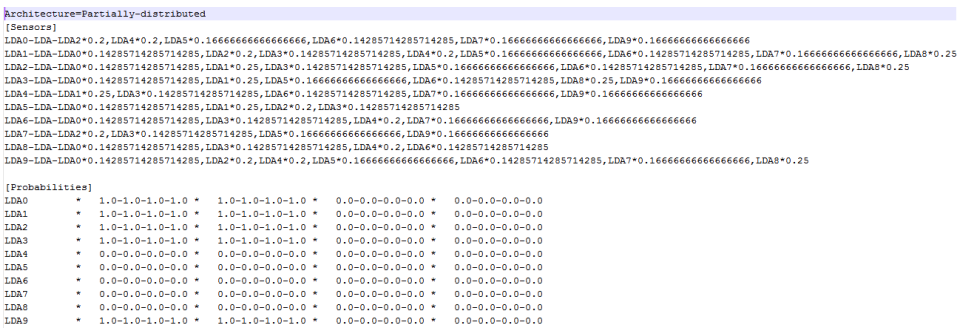

Figura 8.20: Extracto de un fichero .net.

- Fichero spea2.params. El usuario puede editar este fichero para decrementar o incrementar el número de generaciones que realiza el algoritmo SPEA2 en el proceso de generación de una solución (véase el punto *8.2 11.4. Generación de soluciones óptimas*). Se debe tener en cuenta que un valor elevado puede ralentizar el equipo y demorar la operación por un largo período de tiempo.